

U.S. Department of Defense

**High Level Architecture
Interface Specification
Version 1.3**

DRAFT 9

5 February 1998

Contents

1. Overview.....	1
1.1 Scope.....	1
1.2 Purpose	1
1.3 Background	1
1.3.1 HLA federation object model framework	1
1.3.2 General nomenclature and conventions.....	2
1.3.3 Organization of this document.....	3
2. References.....	4
3. Abbreviations, acronyms, and definitions	4
3.1 Abbreviations and acronyms	4
3.2 Definitions	4
4. Federation management	5
4.1 Overview	5
4.2 Create Federation Execution	13
4.3 Destroy Federation Execution	14
4.4 Join Federation Execution	15
4.5 Resign Federation Execution	16
4.6 Register Federation Synchronization Point	17
4.7 Confirm Synchronization Point Registration †	18
4.8 Announce Synchronization Point †	19
4.9 Synchronization Point Achieved.....	20
4.10 Federation Synchronized †	21
4.11 Request Federation Save	22
4.12 Initiate Federate Save †	23
4.13 Federate Save Begun	24
4.14 Federate Save Complete.....	25
4.15 Federation Saved †	26
4.16 Request Federation Restore	27
4.17 Confirm Federation Restoration Request †	28
4.18 Federation Restore Begun †	29
4.19 Initiate Federate Restore †	30
4.20 Federate Restore Complete	31
4.21 Federation Restored †	32

5. Declaration management.....	33
5.1 Overview	33
5.1.1 Static properties of the FED.....	33
5.1.2 Definitions and constraints for object classes and class attributes	33
5.1.3 Definitions and constraints for interaction classes and parameters	35
5.2 Publish Object Class	43
5.3 Unpublish Object Class.....	45
5.4 Publish Interaction Class	47
5.5 Unpublish Interaction Class.....	48
5.6 Subscribe Object Class Attributes	49
5.7 Unsubscribe Object Class.....	51
5.8 Subscribe Interaction Class.....	52
5.9 Unsubscribe Interaction Class.....	54
5.10 Start Registration For Object Class †	55
5.11 Stop Registration For Object Class †.....	56
5.12 Turn Interactions On †	57
5.13 Turn Interactions Off †.....	58
6. Object management.....	59
6.1 Overview	59
6.2 Register Object Instance	64
6.3 Discover Object Instance †	65
6.4 Update Attribute Values	66
6.5 Reflect Attribute Values †.....	67
6.6 Send Interaction	68
6.7 Receive Interaction †	69
6.8 Delete Object Instance	70
6.9 Remove Object Instance †.....	71
6.10 Local Delete Object Instance.....	72
6.11 Change Attribute Transportation Type.....	73
6.12 Change Interaction Transportation Type.....	74
6.13 Attributes In Scope †	75
6.14 Attributes Out Of Scope †.....	76
6.15 Request Attribute Value Update.....	77
6.16 Provide Attribute Value Update †	78
6.17 Turn Updates On For Object Instance †	79
6.18 Turn Updates Off For Object Instance †.....	80
7. Ownership management.....	81
7.1 Overview	81
7.1.1 Ownership and publication	84
7.1.2 Ownership transfer	84
7.1.3 PriviledgeToDeleteObject	87
7.1.4 User-supplied tags	87
7.1.5 Sets of attribute designators	87
7.2 Unconditional Attribute Ownership Divestiture	88
7.3 Negotiated Attribute Ownership Divestiture	89

7.4 Request Attribute Ownership Assumption †	91
7.5 Attribute Ownership Divestiture Notification †	92
7.6 Attribute Ownership Acquisition Notification †	93
7.7 Attribute Ownership Acquisition	94
7.8 Attribute Ownership Acquisition If Available	96
7.9 Attribute Ownership Unavailable †	97
7.10 Request Attribute Ownership Release †	98
7.11 Attribute Ownership Release Response	99
7.12 Cancel Negotiated Attribute Ownership Divestiture	100
7.13 Cancel Attribute Ownership Acquisition	101
7.14 Confirm Attribute Ownership Acquisition Cancellation †	103
7.15 Query Attribute Ownership	104
7.16 Inform Attribute Ownership †	105
7.17 Is Attribute Owned By Federate	106
8. Time management	107
8.1 Overview	107
8.1.1 Messages	107
8.1.2 Logical time	109
8.1.3 Time-regulating federates	109
8.1.4 Time-constrained federates	110
8.1.5 Advancing time	110
8.1.6 Putting it all together	112
8.2 Enable Time Regulation	116
8.3 Time Regulation Enabled †	118
8.4 Disable Time Regulation	119
8.5 Enable Time Constrained	120
8.6 Time Constrained Enabled †	121
8.7 Disable Time Constrained	122
8.8 Time Advance Request	123
8.9 Time Advance Request Available	125
8.10 Next Event Request	127
8.11 Next Event Request Available	129
8.12 Flush Queue Request	131
8.13 Time Advance Grant †	133
8.14 Enable Asynchronous Delivery	135
8.15 Disable Asynchronous Delivery	136
8.16 Query LBTS	137
8.17 Query Federate Time	138
8.18 Query Minimum Next Event Time	139
8.19 Modify Lookahead	140
8.20 Query Lookahead	141
8.21 Retract	142
8.22 Request Retraction †	143
8.23 Change Attribute Order Type	144
8.24 Change Interaction Order Type	145
9. Data distribution management	147

9.1 Overview	147
9.2 Create Region.....	151
9.3 Modify Region.....	152
9.4 Delete Region	153
9.5 Register Object Instance With Region.....	154
9.6 Associate Region For Updates	156
9.7 Unassociate Region For Updates.....	158
9.8 Subscribe Object Class Attributes With Region.....	159
9.9 Unsubscribe Object Class With Region	161
9.10 Subscribe Interaction Class With Region	162
9.11 Unsubscribe Interaction Class With Region	164
9.12 Send Interaction With Region.....	165
9.13 Request Attribute Value Update With Region	167
9.14 Change Thresholds †	168
10. Support services	169
10.1 Overview	169
10.2 Get Object Class Handle	170
10.3 Get Object Class Name	171
10.4 Get Attribute Handle	172
10.5 Get Attribute Name	173
10.6 Get Interaction Class Handle	174
10.7 Get Interaction Class Name	175
10.8 Get Parameter Handle.....	176
10.9 Get Parameter Name.....	177
10.10 Get Object Instance Handle.....	178
10.11 Get Object Instance Name.....	179
10.12 Get Routing Space Handle	180
10.13 Get Routing Space Name.....	181
10.14 Get Dimension Handle.....	182
10.15 Get Dimension Name	183
10.16 Get Attribute Routing Space Handle	184
10.17 Get Object Class	185
10.18 Get Interaction Routing Space Handle	186
10.19 Get Transportation Handle.....	187
10.20 Get Transportation Name	188
10.21 Get Ordering Handle	189
10.22 Get Ordering Name	190
10.23 Enable Class Relevance Advisory Switch.....	191
10.24 Disable Class Relevance Advisory Switch.....	192
10.25 Enable Attribute Relevance Advisory Switch	193
10.26 Disable Attribute Relevance Advisory Switch	194
10.27 Enable Attribute Scope Advisory Switch.....	195
10.28 Disable Attribute Scope Advisory Switch.....	196
10.29 Enable Interaction Relevance Advisory Switch	197
10.30 Disable Interaction Relevance Advisory Switch	198
11. Management object model (MOM).....	199

11.1 MOM objects.....	201
11.1.1 Object class <i>Manager.Federation</i>	201
11.1.2 Object class <i>Manager.Federate</i>	202
11.2 MOM interactions	203
11.2.1 Interaction class <i>Manager.Federate.Adjust</i>	204
11.2.2 Interaction class <i>Manager.Federate.Request</i>	205
11.2.3 Interaction class <i>Manager.Federate.Report</i>	209
11.2.4 Interaction class <i>Manager.Federate.Service</i>	216
12. Federation execution data (FED)	229
12.1 FED data interchange format (FED DIF)	229
12.1.1 BNF notation of the DIF	229
12.1.2 HLA FED DIF BNF definition.....	231
12.1.3 FED DIF meta-data consistency.....	232
12.1.4 FED DIF glossary	233
12.2 Example FED file	233
ANNEX A (normative) IDL application programmer's interface.....	243
ANNEX B (normative) C++ application programmer's interface	245
ANNEX C (normative) Ada 95 application programmer's interface.....	289
ANNEX D (normative) Java application programmer's interface	291
ANNEX E (informative) Bibliography	293

List of Figures

Figure 1—Basic states of the federation execution.....	5
Figure 2—Overall view of federate-to-RTI relationship.....	6
Figure 3—Lifetime of a federate.....	9
Figure 4—Normal activity.....	11
Figure 5—Object class (i).....	38
Figure 6—Class attribute (i).....	40
Figure 7—Interaction class (i).....	42
Figure 8—Object instance (i) known	61
Figure 9—Instance attribute (i)	62
Figure 10—Implications of ownership of instance attribute (i)	63
Figure 11—Establishing ownership of instance attribute (i).....	82
Figure 12—Temporal state.....	114
Figure 13—Routing space of two dimensions.....	150
Figure 14—MOM object class structure	199
Figure 15—MOM interaction class structure.....	200
Figure 16—Basic BNF constructs.....	230
Figure 17—HLA FED DIF BNF definition.....	232
Figure 18—FED file with MOM definitions.....	234

List of Tables

Table 1—Order type of a sent message	108
Table 2—Order type of a received message	109
Table 3—Service descriptions	111
Table 4—Object class <i>Manager.Federation</i>	201
Table 5—Object class <i>Manager.Federate</i>	202
Table 6—Interaction subclass <i>SetTiming</i>	204
Table 7—Interaction subclass <i>ModifyAttributeStat</i>	204
Table 8—Interaction subclass <i>SetServiceReporting</i>	205
Table 9—Interaction subclass <i>SetExceptionLogging</i>	205
Table 10—Interaction subclass <i>RequestPublications</i>	205
Table 11—Interaction subclass <i>RequestSubscriptions</i>	206
Table 12—Interaction subclass <i>RequestObjectsOwned</i>	206
Table 13—Interaction subclass <i>RequestObjectsUpdated</i>	206
Table 14—Interaction subclass <i>RequestObjectsReflected</i>	206
Table 15—Interaction subclass <i>RequestUpdatesSent</i>	208
Table 16—Interaction subclass <i>RequestInteractionsSent</i>	208
Table 17—Interaction subclass <i>RequestReflectionsReceived</i>	208
Table 18—Interaction subclass <i>RequestInteractionsReceived</i>	209
Table 19—Interaction subclass <i>RequestObjectInformation</i>	209
Table 20—Interaction subclass <i>ReportObjectPublication</i>	210
Table 21—Interaction subclass <i>ReportInteractionPublication</i>	210
Table 22—Interaction subclass <i>ReportObjectSubscription</i>	210
Table 23—Interaction subclass <i>ReportInteractionSubscription</i>	211
Table 24—Interaction subclass <i>ReportObjectsOwned</i>	211
Table 25—Interaction subclass <i>ReportObjectsUpdated</i>	212
Table 26—Interaction subclass <i>ReportObjectsReflected</i>	212
Table 27—Interaction subclass <i>ReportUpdatesSent</i>	212
Table 28—Interaction subclass <i>ReportReflectionsReceived</i>	213
Table 29—Interaction subclass <i>ReportInteractionsSent</i>	213
Table 30—Interaction subclass <i>ReportInteractionsReceived</i>	214
Table 31—Interaction subclass <i>ReportObjectInformation</i>	214
Table 32—Interaction subclass <i>Alert</i>	214
Table 33—Interaction subclass <i>ReportServiceInvocation</i>	215
Table 34—Interaction subclass <i>ResignFederationExecution</i>	217
Table 35—Interaction subclass <i>InitiatePause</i>	217
Table 36—Interaction subclass <i>PauseAchieved</i>	217
Table 37—Interaction subclass <i>InitiateFederateSave</i>	218
Table 38—Interaction subclass <i>FederateSaveBegun</i>	218

Table 39—Interaction subclass <i>FederateSaveComplete</i>	219
Table 40—Interaction subclass <i>InitiateFederateRestore</i>	219
Table 41—Interaction subclass <i>FederateRestoreComplete</i>	219
Table 42—Interaction subclass <i>PublishObjectClass</i>	220
Table 43—Interaction subclass <i>UnpublishObjectClass</i>	220
Table 44—Interaction subclass <i>PublishInteractionClass</i>	220
Table 45—Interaction subclass <i>UnpublishInteractionClass</i>	220
Table 46—Interaction subclass <i>SubscribeObjectClassAttributes</i>	221
Table 47—Interaction subclass <i>UnsubscribeObjectClass</i>	221
Table 48—Interaction subclass <i>SubscribeInteractionClass</i>	221
Table 49—Interaction subclass <i>UnsubscribeInteractionClass</i>	223
Table 50—Interaction subclass <i>DeleteObjectInstance</i>	223
Table 51—Interaction subclass <i>LocalDeleteObjectInstance</i>	223
Table 52—Interaction subclass <i>ChangeAttributeTransportationType</i>	224
Table 53—Interaction subclass <i>ChangeAttributeOrderType</i>	224
Table 54—Interaction subclass <i>ChangeInteractionTransportationType</i>	224
Table 55—Interaction subclass <i>ChangeInteractionOrderType</i>	225
Table 56—Interaction subclass <i>UnconditionalAttributeOwnershipDivestiture</i>	225
Table 57—Interaction subclass <i>EnableTimeRegulation</i>	226
Table 58—Interaction subclass <i>DisableTimeRegulation</i>	226
Table 59—Interaction subclass <i>EnableTimeConstrained</i>	226
Table 60—Interaction subclass <i>DisableTimeConstrained</i>	226
Table 61—Interaction subclass <i>EnableAsynchronousDelivery</i>	227
Table 62—Interaction subclass <i>EnableAsynchronousDelivery</i>	227
Table 63—Interaction subclass <i>ModifyLookahead</i>	227
Table 64—Interaction subclass <i>TimeAdvanceRequest</i>	227
Table 65—Interaction subclass <i>TimeAdvanceRequestAvailable</i>	228
Table 66—Interaction subclass <i>NextEventRequest</i>	228
Table 67—Interaction subclass <i>NextEventRequestAvailable</i>	228
Table 68—Interaction subclass <i>FlushQueueRequest</i>	228

1. Overview

1.1 Scope

The formal definition of the Department of Defense High Level Architecture (HLA) comprises three main components: the HLA rules, the HLA interface specification, and the HLA object model template (OMT) (see Clause 2). This document shall provide a complete description of the essential elements of the second component of the HLA, the interface specification. The other two components of the HLA formal definition are described in the following documents:

- *High Level Architecture, Rules, Version 1.3*
- *High Level Architecture, Object Model Template, Version 1.3*

In addition to these reference documents, the HLA technical library on the DMSO homepage (<http://www.dmsol.mil>) contains other information sources relevant to developing and executing HLA federations.

1.2 Purpose

This document provides a specification for the DoD HLA functional interfaces between federates and the runtime infrastructure (RTI). The RTI provides services to federates in a way that is analogous to how a distributed operating system provides services to applications. These interfaces are arranged into six basic RTI service groups:

- a) federation management
- b) declaration management
- c) object management
- d) ownership management
- e) time management
- f) data distribution management

The six service groups describe the interface between the federates and the RTI, and the software services provided by the RTI for use by HLA federates. The initial set of these services was carefully chosen to provide those functions most likely to be required across multiple federations. As a result, federate applications will require most of the services described in this document. The RTI requires a set of services from the federate that are referred to as “RTI initiated” and are denoted with a †.

1.3 Background

1.3.1 HLA federation object model framework

A concise and rigorous description of the object model framework is essential to the specification of the interface between federates and the RTI and of the RTI services. The rules and terminology used to describe a federation object model (FOM) are described in the *High Level Architecture, Object Model Template, Version 1.3*. A simulation object model (SOM) describes salient characteristics of a federate to aid in its reuse and other activities focused on the details of its internal operation. As such, a SOM is not the concern of the RTI and its services. A FOM, on the other hand, deals with inter-federate issues and is relevant to the use of the RTI. The DoD HLA definition states that FOMs describe

- The set of object classes chosen to represent the real world for a planned federation,
- The set of interaction classes chosen to represent the interplay among real-world objects,
- The attributes and parameters of these classes, and
- The level of detail at which these classes represent the real world, including all characteristics.

Every object is an instance of an object class found in the FOM. Object classes are chosen by the object model designer to facilitate a desired organizational scheme. Each object class has a set of attributes associated with it. An

attribute is a distinct, identifiable portion of the object state. In this discussion, “attribute designator” refers to the attribute and “attribute value” refers to its contents. From the federation perspective, the set of all attribute values for an object instance shall completely define the state of the instance. Federates may associate additional state information with an object instance that is not communicated between federates, but this is outside the purview of the HLA federation object model.

Federates use the state of the object instances as one of the primary means of communication. At any time, only one federate shall be responsible for simulating a given object instance attribute. That federate shall provide new values for that instance attribute to the other federates in the federation execution through the RTI services. The federate providing the new instance attribute values shall be said to be *updating* that instance attribute value. Federates receiving those values shall be said to be *reflecting* that instance attribute.

The privilege to update a value for an instance attribute shall be uniquely held by a single federate at any given time during a federation execution. A federate that has the privilege to update values for an instance attribute shall be said to *own* that instance attribute. The RTI provides services that allow federates to exchange ownership of object instance attributes. The federate that registers an object instance shall automatically own the *privilegeToDeleteObject* instance attribute for that instance (all federates automatically publish the *privilegeToDeleteObject* for all object classes they explicitly publish). The RTI provides services that allow federates to transfer the “*privilegeToDeleteObject*” attribute in the same way as other attributes.

Each object instance shall have a designator. The value of an object instance designator shall be unique for each federation execution. Object instance designators shall be dynamically generated by the RTI.

The FOM framework also allows for interaction classes for each object model. The types of interactions possible and their parameters are specified within the FOM.

A *federation* is the combination of a particular FOM, a particular set of federates, and the RTI services. A federation is designed for a specific purpose using a commonly understood federation object model and a set of federates that can associate their individual semantics with that object model. A *federation execution* is an instance of the *Create Federation Execution* service invocation and entails executing the federation with a specific FOM and an RTI, and using various execution details.

1.3.2 General nomenclature and conventions

There are various entities (classes, attributes, parameters, regions, federates, object instances, etc.) referenced in this document that can have the following different “views”:

- *Name* - human readable or for communication between federates
- *Handle* - capable of being manipulated by a computer or for communication between a federate and the RTI

The arguments to the services described in this document will use different views of the entities depending on a particular RTI implementation. For clarity, this document refers to only a generic view, known as a “designator,” when referring to these entities.

The following sets of data are needed for the implementation of a running RTI and federation executions:

- *Federation execution data (FED)* - information derived from the FOM (class, attribute, parameter names, etc.) and used by the RTI at runtime. Each federation execution needs one. In the abstract, creation of a federation execution is simply the binding of a federation execution name to a FED. The organization of FEDs will become the subject of standardization so that object model development tools can automatically generate them for any vendor’s RTI.
- *RTI initialization data (RID)* - RTI vendor-specific information needed to run an RTI. A RID is probably supplied when an RTI is initialized.

For all federate-initiated services in this specification, except 4.2 Create Federation Execution, 4.3 Destroy Federation Execution, and 4.4 Join Federation Execution, there is an implied supplied argument that is a federate’s connection to a federation execution. For all RTI-initiated services, there is an implied supplied argument that is also a federate’s connection to a federation execution. The manner in which these arguments are actually provided to the services is dependent on the RTI implementation, and therefore is not shown in the service descriptions.

Also, for the RTI-initiated services there are some implicit pre-conditions which are not stated explicitly because the RTI is assumed to be well-behaved.

1.3.3 Organization of this document

The six HLA service groups and support services are specified in Clauses 4 through 10. Each service is described using several components:

- Name & Description - service name and narrative describing the functionality of the service
- Supplied Arguments - required and optional service-initiator provided arguments
- Returned Arguments - arguments returned by the service
- Pre-conditions - conditions that must exist for the service to execute correctly
- Post-conditions - conditions that will exist once the service has executed correctly
- Exceptions - notifications of any irregularity that may occur during service execution
- Related Services - other HLA services that are related to this service

After the clauses describing the service groups are clauses describing the management object model (MOM) and federation execution data (FED).

Annexes A through D contain HLA application programmer's interfaces (APIs) for the following languages:

- IDL
- C++
- Ada 95
- Java.

NOTE—Comments on this document should be sent by electronic mail to the Defense Modeling and Simulation Office (DMSO) HLA specifications e-mail address (hla_specs@msis.dmsi.mil). The subject line of the message should include the clause number referenced in the comment. The body of each submittal should include: (1) the name and e-mail address of the person making the comment (separate from the mail header), (2) reference to the RTI service or portion of the I/F specification that the comment addresses (by clause and page number), (3) a one-sentence summary of the comment/issue, (4) a brief description of the comment/issue, and (5) any suggested resolution or action to be taken.

2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply:

U.S. Department of Defense, *High Level Architecture, Rules, Version 1.3*,

U.S. Department of Defense, *High Level Architecture, Object Model Template, Version 1.3*,

3. Abbreviations, acronyms, and definitions

3.1 Abbreviations and acronyms

API	application programmer's interface
BNF	Backus-Naur Form
DIF	data interchange format
DoD	Department of Defense
DMSO	Defense Modeling and Simulation Office
EBNF	extended Backus-Naur Form
FED	federation execution data
FOM	federation object model
HLA	High Level Architecture
LBTS	lower bound on the time stamp
LHS	left-hand side
MOM	management object model
OMDT	object model development tool
RHS	right-hand side
RID	RTI initialization data
RO	receive order
RTI	runtime infrastructure
SOM	simulation object model
TBS	to be supplied
TSO	time-stamped order

3.2 Definitions

For the purposes of this standard, the following terms and definitions shall apply:

TBS

4. Federation management

4.1 Overview

“Federation management” shall refer to the creation, dynamic control, modification, and deletion of a federation execution.

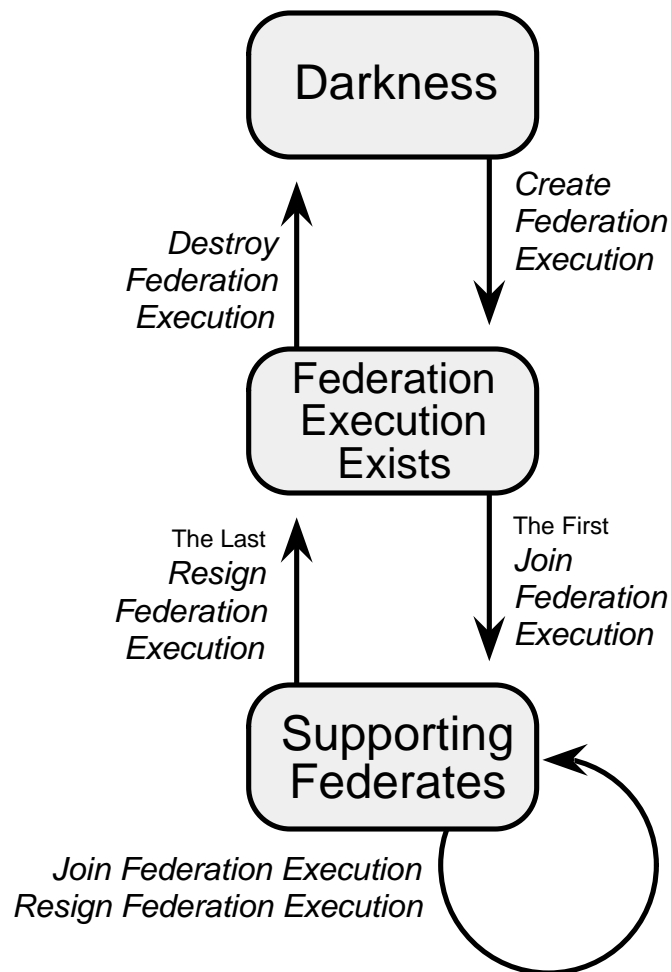


Figure 1—Basic states of the federation execution

Before a federate can join a federation execution, the federation execution must exist. Figure 1 shows the overall state of a federation execution as certain basic federation management services are employed.

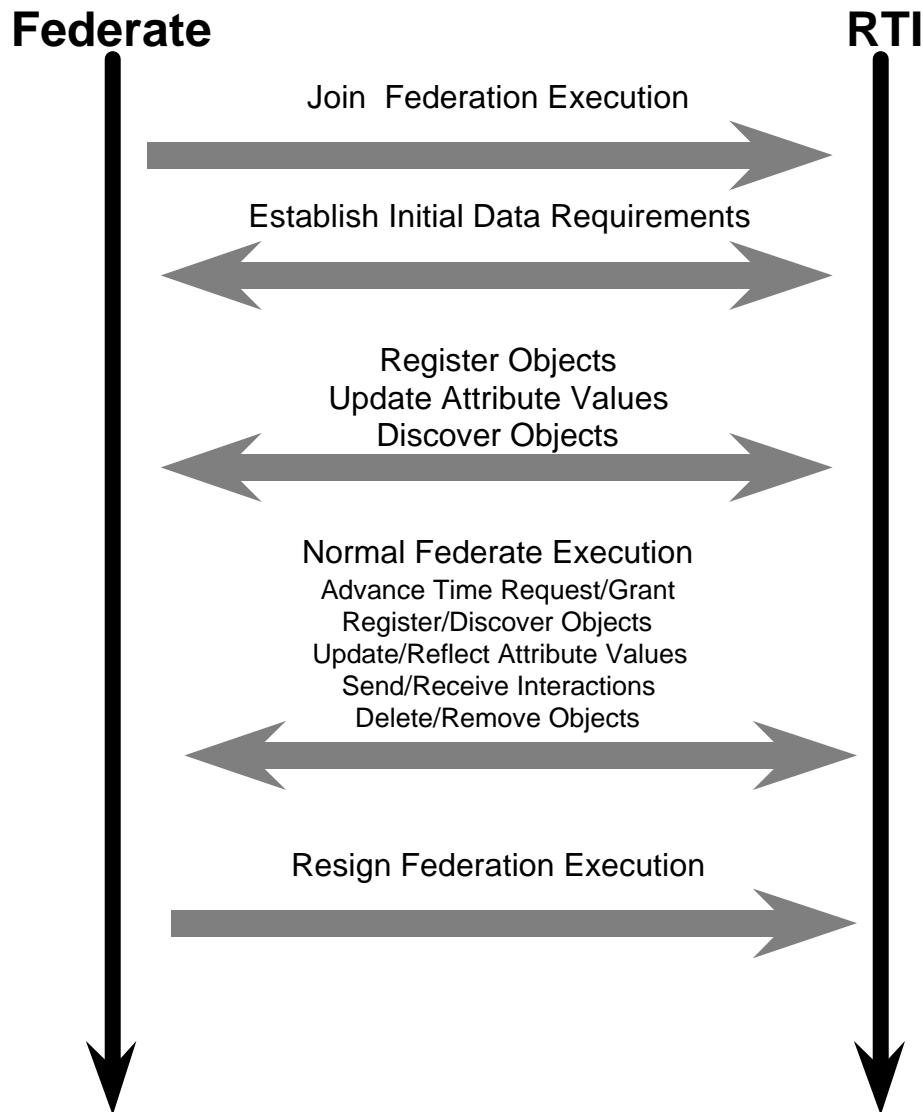


Figure 2—Overall view of federate-to-RTI relationship

Once a federation execution exists, federates may join and resign from it in any sequence that is meaningful to the federation user. Figure 2 presents a generalized view of the basic relationship between a federate and the RTI during the federate participation in a federation execution. The broad arrows in Figure 2 represent the general invocation of RTI service groups and are not intended to demonstrate strict ordering requirements on the use of the services. The HLA concept shall not preclude a single software system from participating in a federation execution as multiple federates nor shall it preclude a given system from participating in multiple (independent) federation executions.

The state diagram in Figure 3 is the first of a series of hierarchical state diagrams that formally describe the state of a given federate, from the perspective of that federate, in varying levels of detail. These state diagrams are formal, accurate descriptions of federate state information depicted in the highly structured,

compact, and expressive *statechart* notation pioneered by David Harel [A1]. In the next few paragraphs we describe the first two of these statecharts in detail as a way of introducing some of Harel's notation and providing an understanding of how the complete set of statecharts in this document are hierarchically interrelated.

As shown in Figure 3, with the successful completion of the *Join Federation Execution* service, a federate shall be in the Joined Federate state, where it shall remain until it resigns from the federation execution. As indicated by the dashed line in the Joined Federate state, the Joined Federate state shall consist of two parallel state machines: one having to do with whether or not the federate is in the process of saving or restoring federate state (depicted to the left of the dashed line), and the other having to do with whether or not the federate is permitted to perform normal activity (depicted to the right of the dashed line). While in the Joined Federate state, the federate shall simultaneously be in both a state depicted in the state machine to the left of the dashed line and a state depicted in the state machine to the right of the dashed line. Initially, upon entering the Joined Federate state, the federate shall be in the Active and Normal Activity Permitted states, as indicated by the dark-circle start transitions. There are interdependencies between these two parallel state machines and between the state machine on the left and the Temporal state machine that appears later in this document. These interdependencies are depicted by the guards (shown within square brackets) that are associated with some state transitions. If a transition has a guard associated with it, the assertion within the guard must be true in order for a federate to make the associated transition from one state to another.

As an example of an interdependency between the two parallel state machines depicted in the Joined Federate state, if a federate that is in the Active state receives a *Federation Restore Begun* † service invocation, it will transition into the Prepared to Restore state (as indicated by the label on the transition from the Active state to the Prepared to Restore state). Once the federate enters the Prepared to Restore state, it also enters the Normal Activity Not Permitted state (as indicated by the guard on the transition from the Normal Activity Permitted to the Normal Activity Not Permitted state). That is, the guards impose the following constraints on a federate: A federate may be in the Normal Activity Permitted state (right side) if and only if it is also in the Active state (left side); and a federate may be in the Normal Activity Not Permitted state (right side) if and only if it is also in the Instructed to Save, Saving, Waiting for Federation to Save, Prepared to Restore, Restoring, or Waiting for Federation to Restore state (left side).

The interdependency between the state machine on the left and the Temporal state machine depicted later in this document is this: a federate that is in the Active state shall not receive an invocation of the *Initiate Federate Save* † service unless that federate is either in the *Not Constrained* or the *Time Advancing* state. The *Not Constrained* and *Time Advancing* states are depicted in the Temporal State diagram in Figure 12. The fact that these two time management related states are mentioned in the guard on the transition from the Active to the Instructed to Save state demonstrates the interdependencies between a federate's save/restore state and its temporal state. Specifically, it indicates that a federate must either be not constrained by time management or be in a position to receive a time advance grant in order for it to receive an invocation of the *Initiate Federate Save* † service.

If a federate is in the Normal Activity Permitted state, the federate may perform normal federate activity such as registering and discovering object instances, publishing and subscribing to object class attributes and interactions, updating and reflecting instance attribute values, sending and receiving interactions, deleting and removing object instances, and requesting or receiving time advance grants. The Normal Activity Permitted state, simple as it may appear in the Joined Federate statechart, actually contains all of the other states that appear in the statecharts that appear subsequently in this document. Together, these statecharts formally describe the state of a federate from that federate's perspective. These statecharts are

complete in the sense that all transitions shown represent legal operations and transitions that are not shown represent illegal operations. Illegal operations shall generate exceptions if invoked.

PLACE

Figure 3—Lifetime of a federate

HERE

The Normal Activity Permitted state depicted in Figure 3 is elaborated in further detail in Figure 4 to identify the three major portions of federate state: time management (indicated by the Temporal state), state associated with each object class, and state associated with each interaction class. When a federate enters the Joined Federate state, the federate shall have a temporal state and object and interaction class states. The federate shall have an Object Class state for each object class that is defined in the FED file that is associated with the federate execution. Likewise, the federate shall have an Interaction Class state for each interaction class that is defined in the FED. A federate shall be in the temporal state and in each of these object and interaction class states simultaneously (as depicted by the dashed lines separating the state machines within the Temporal state). Time management is elaborated in further detail in Figure 12, the Temporal State statechart. The state of an arbitrary object class is elaborated in further detail in Figure 5, the Object Class (i) statechart, and the state of an arbitrary interaction class is elaborated in further detail in Figure 7, the Interaction Class (i) statechart.

PLACE

Figure 4—Normal activity

HERE

As depicted in Figure 4, a federate that has joined a federation execution may be in one of the many states. Any federate in the execution may initiate a save by invoking the *Request Federation Save* service. If there is no federation time argument provided with the invocation of this service, the RTI shall instruct all of the federates in the federation execution (including the requesting federate) to save state by invoking the *Initiate Federate Save* † service at all of these federates as soon as possible. If there is a federation time argument provided, the RTI shall invoke the *Initiate Federate Save* † service at each of the time-constrained federates when their value of logical time advances to the value provided, and it shall invoke the *Initiate Federate Save* † service at all non-time-constrained federates as soon as possible after it has invoked it at all of the time-constrained federates.

When a federate receives an *Initiate Federate Save* † service invocation and subsequently saves its state, it shall use the federation save label (which was specified by the federate requesting the save in the *Request Federation Save* service) and its federate type (which it specified when it joined the federation execution) to distinguish the saved information. The saved information shall be persistent, meaning that it is stored onto disk or some other persistent medium and it remains intact even after the federation execution is destroyed. The saved information can thus be used at a later date, by some new set of federates, to restore all federates in the federation execution to the state that they were in when the save was accomplished. The federation can then resume execution of the simulation from that saved point. The set of federates joined to an execution when state is restored from a previously saved state need not be the exact set of federates that were joined to the federation execution when the state being restored was saved. The number of federates of each federate type that are joined to the federation execution, however, shall be the same. The federate-type parameter argument supplied in the *Join Federation Execution* service invocation, therefore, is crucial to the save-and-restore process. Declaring a federate to be of a given type is equivalent to asserting that the federate can be restored using the state information saved by any other federate of that type.

4.2

4.2 Create Federation Execution

The *Create Federation Execution* service shall create a new federation execution and add it to the set of supported federation executions. Each federation execution created by this service shall be independent of all other federation executions, and there shall be no inter-communication within the RTI between federation executions. The FED designator argument shall identify the location of FED data that is required for the federation execution to be created.

Supplied Arguments

- Federation execution name
- FED designator (see Clause 12)

Returned Arguments

- None

Pre-conditions

- The federation execution does not exist.

Post-conditions

- A federation execution exists with the given name that can be joined by federates.

Exceptions

- The federation execution already exists.
- Could not locate FED information from supplied designator
- Invalid FED
- RTI internal error

Related Services

- *Destroy Federation Execution*

4.3 Destroy Federation Execution

The *Destroy Federation Execution* service shall remove a federation execution from the RTI set of supported federation executions. All federation activity shall have stopped and all federates shall have resigned before invoking this service.

Supplied Arguments

- Federation execution name

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- There are no federates joined to this federation execution.

Post-conditions

- The federation execution does not exist.

Exceptions

- Federates are joined to the federation execution.
- The federation execution does not exist.
- RTI internal error

Related Services

- *Create Federation Execution*

4.4 Join Federation Execution

The *Join Federation Execution* service shall affiliate the federate with a federation execution. Invocation of the *Join Federation Execution* service shall indicate the intention to participate in the specified federation. The federate type parameter shall distinguish federate categories for federation save-and-restore purposes. The returned federate designator shall be unique across all federates in a federation execution.

Supplied Arguments

- Federate type
- Federation execution name

Returned Arguments

- Federate designator

Pre-conditions

- The federation execution exists.
- The federate is not joined to that execution.

Post-conditions

- The federate is a member of the federation execution.

Exceptions

- The federate is already joined to the federation execution.
- The specified federation execution does not exist.
- Invalid FED
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Resign Federation Execution*
- *Request Restore*

4.5 Resign Federation Execution

The *Resign Federation Execution* service shall indicate the requested cessation of federation participation. Before resigning, ownership of instance attributes held by the federate should be resolved. The federate can transfer ownership of these instance attributes to other federates, release them for ownership acquisition at a later time, or delete the object instance of which they are a part (assuming the federate has the privilege to delete these object instances). As a convenience to the federate, the *Resign Federation Execution* service shall accept an action argument that directs the RTI to perform zero or more of the following actions:

- Release all owned instance attributes for future ownership acquisition. This shall place the instance attributes into an unowned state (implying that their values are not being updated), which shall make them eligible for ownership by another federate. See Clause 7 for a more detailed description.
- Delete all object instances for which the federate has that privilege (implied invocation of the *Delete Object Instance* service).

Supplied Arguments

- Directive to
 - a) release ownership of all owned instance attributes
 - b) delete all object instances for which the federate has the delete privilege
 - c) perform action (a) and then action (b)
 - d) perform no actions

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- If directive (b) is supplied, the federate does not own any instance attributes of object instances for which it does not also have the delete privilege.
- If directive (d) is supplied, the federate does not own any instance attributes in the federation execution.

Post-conditions

- The federate is not a member of the federation execution.
- There are no instance attributes in the federation execution owned by the federate.
- If directive (b) or (c) are supplied all object instances for which the federate has the delete privilege are deleted.

Exceptions

- The federate owns instance attributes.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Join Federation Execution*

4.6 Register Federation Synchronization Point

The *Register Federation Synchronization Point* service shall be used to initiate the registration of an upcoming synchronization point label. When a synchronization point label has been successfully registered (indicated through the *Confirm Synchronization Point Registration* † service) the RTI shall inform some or all federates of the label existence by invoking the *Announce Synchronization Point* † service at those federates. The optional set of federate designators shall be used by the federate to specify which federates in the execution should be informed of the label existence. If the optional set of federate designators is empty or not supplied, all federates in the federation execution shall be informed of the label existence. If the optional set of designators is not empty, all designated federates must be federation execution members. The user-supplied tag shall provide a vehicle for information to be associated with the synchronization point and shall be announced along with the synchronization label. It is possible for multiple synchronization points registered by the same or different federates to be pending at the same time. The synchronization labels, however, shall be unique.

Supplied Arguments

- Synchronization point label
- User-supplied tag
- Optional set of federate designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- If an optional set of federate designators is supplied, those federates must have joined the federation execution.

Post-conditions

- The synchronization label is known to the RTI.

Exceptions

- The federate not a federation execution member.
- The designated federate not a federation execution member.
- The synchronization label is not unique.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Confirm Synchronization Point Registration* †
- *Announce Synchronization Point* †\

4.7 Confirm Synchronization Point Registration †

The *Confirm Synchronization Point Registration* † service shall indicate to the federate the status of a requested federation synchronization point registration. This service shall be invoked in response to a *Register Federation Synchronization Point* service invocation. A positive success indicator informs the federate that the label has been successfully registered. A negative success indicator informs the federate that the label was already in use or that the registration of this label has otherwise failed. A registration attempt that ends with a negative success indicator shall have no other effect on the federation execution.

Supplied Arguments

- Synchronization point label
- Registration success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has invoked *Register Federation Synchronization Point* service for the specified label.

Post-conditions

- If the registration success indicator is positive, the specified label and associated user supplied tag will be announced to the appropriate federates.
- If the registration success indicator is negative, this service and the corresponding *Register Federation Synchronization Point* service invocation have no consequence.

Exceptions

- Federate internal error.

Related Services

- *Register Federation Synchronization Point*

4.8 Announce Synchronization Point †

The *Announce Synchronization Point †* service shall inform a federate of the existence of a new synchronization point label. When a synchronization point label has been registered with the *Register Federation Synchronization Point* service, the RTI shall invoke the *Announce Synchronization Point †* service at either all or at the specified set of federates to inform them of the label existence. The federates informed of the existence of a given synchronization point label via the *Announce Synchronization Point †* service shall form the synchronization set for that point. If the optional set of federate designators was null or not provided when the synchronization point label was registered, the RTI shall invoke the *Announce Synchronization Point †* service at all federates that join the federation execution after the synchronization label was registered but before all federates that were informed of the synchronization label existence have invoked the *Synchronization Point Achieved* service. These newly joining federates shall also become part of the synchronization set for that point. Federates that resign from the federation execution after the announcement of a synchronization point but before the federation synchronizes at that point shall be removed from the synchronization set. The user-supplied tag supplied by the *Announce Synchronization Point †* service shall be the tag that was supplied to the corresponding *Register Federation Synchronization Point* service invocation.

Supplied Arguments

- Synchronization point label
- User-supplied tag

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The synchronization point has been registered.

Post-conditions

- The synchronization label is known to the federate and can be used in the *Synchronization Point Achieved* and *Federation Synchronized †* services.

Exceptions

- Federate internal error

Related Services

- *Register Federation Synchronization Point*

4.9 Synchronization Point Achieved

The *Synchronization Point Achieved* service shall inform the RTI that the federate has reached the specified synchronization point. Once the last federate in the synchronization set for a given point has invoked this service, the RTI shall not invoke the *Announce Synchronization Point* † on any newly joining federates.

Supplied Arguments

- Synchronization point label

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The synchronization point has been announced.

Post-conditions

- The federate is noted as having reached the specified synchronization point.

Exceptions

- The synchronization label is not registered.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Federation Synchronized* †

4.10 Federation Synchronized †

The *Federation Synchronized* † service shall inform the federate that all federates in the synchronization set of the specified synchronization point have invoked the *Synchronization Point Achieved* service for that point. This service shall be invoked at all federates that are in the synchronization set for that point, indicating that the federates in the synchronization set have synchronized at that point. Once the synchronization set for a point synchronizes at that point, that point shall no longer be registered and the synchronization set for that point shall no longer exist.

Supplied Arguments

- Synchronization point label

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The synchronization point has been registered.
- The synchronization point has been announced.
- All federates have invoked *Synchronization Point Achieved* using the specified label.

Post-conditions

- The federate is informed that all federates, including it, have invoked *Synchronization Point Achieved* using the specified label.

Exceptions

- Federate internal error

Related Services

- *Synchronization Point Achieved*

4.11 Request Federation Save

The *Request Federation Save* service shall specify that a federation save should take place. If the optional federation time argument is not present, the RTI shall instruct all federation execution members to save state as soon as possible after the invocation of the *Request Federation Save* service. If the optional federation time argument is present, the RTI shall instruct each time-constrained federate to save state when its value of logical time advances to the value provided; and it shall instruct non-time-constrained federates to save state when the last time-constrained federate's value of logical time advances to the value of the optional federation save time provided. The RTI shall notify a federate to save state by invoking the *Initiate Federate Save* † service at that federate. Only one requested save shall be outstanding at a time. A new save request shall replace any outstanding save request. However, a save request cannot happen during a save in progress, which is between the RTI invocation of the *Initiate Federate Save* † service and RTI invocation of the *Federation Saved* † service.

Supplied Arguments

- Federation save label
- Optional value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- Save not in progress

Post-conditions

- A federation save has been requested.
- All previous requested saves are canceled.

Exceptions

- Federation time has already passed (if optional time argument supplied).
- Federation time is invalid (if optional time argument is supplied).
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Constrained*
- *Initiate Federate Save* †
- *Federate Save Begun*
- *Federate Save Complete*
- *Federation Saved* †
- *Request Restore*

4.12 Initiate Federate Save †

The *Initiate Federate Save †* service shall instruct the federate to save state. The federate should save as soon as possible after the invocation of the *Initiate Federate Save †* service. The label provided to the RTI when the save was requested, via the *Request Federation Save* service, shall be supplied to the federate. The federate shall use this label, the name of the federation execution, and its federate type, which it supplied when it invoked the *Join Federation Execution* service, to distinguish the saved state information. If a federate is not time-constrained, it can expect to receive an *Initiate Federate Save †* service invocation at any time. If a federate is time constrained, it can expect to receive an *Initiate Federate Save †* service invocation only when one of the following services is pending: *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request*. The federate shall stop providing new information to the federation immediately after receiving the *Initiate Federate Save †* service invocation. The federate may resume providing new information to the federation only after receiving the *Federation Saved †* service invocation.

Supplied Arguments

- Federation save label

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- A federation save has been scheduled.

Post-conditions

- The federate has been notified to begin saving its state.

Exceptions

- Unable to perform save
- Federate internal error

Related Services

- *Request Federation Save*
- *Federate Save Begun*
- *Federate Save Complete*
- *Federation Saved †*

4.13 Federate Save Begun

The *Federate Save Begun* service shall notify the RTI that the federate is beginning to save its state.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has received an *Initiate Federate Save* † invocation.
- The federate is ready to start saving its state.

Post-conditions

- The RTI has been informed that the federate has begun saving its state.

Exceptions

- Save not initiated
- The federate is not a federation execution member.
- Restore in progress
- RTI internal error

Related Services

- *Request Federation Save*
- *Initiate Federate Save* †
- *Federate Save Complete*
- *Federation Saved* †

4.14 Federate Save Complete

The *Federate Save Complete* service shall notify the RTI that the federate has completed its save attempt. The save-success indicator shall inform the RTI that the federate save either succeeded or failed.

Supplied Arguments

- Federate save-success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has invoked the *Federate Save Begun* service for this save.
- The federate has completed the attempt to save its state.

Post-conditions

- The RTI has been informed of the status of the state save attempt.

Exceptions

- Invalid save-success indicator
- Save not initiated
- The federate is not a federation execution member.
- Restore in progress
- RTI internal error

Related Services

- *Request Federation Save*
- *Initiate Federate Save* †
- *Federate Save Begun*
- *Federation Saved* †

4.15 Federation Saved †

The *Federation Saved †* service shall inform the federate that the federation save process is complete, and shall indicate whether it completed successfully or not. If the save-success indicator argument indicates success, this means that all federates at which the *Initiate Federate Save †* service was invoked have invoked the *Federate Save Complete* service with a save-success indicator that indicated success. If the save-success indicator argument indicates failure, this means that one or more federates at which the *Initiate Federate Save †* service was invoked have invoked the *Federate Save Complete* service with a save-success indicator that indicated failure, or that the RTI detected failure at one or more of these federates. All federates that received an invocation of the *Initiate Federate Save †* service shall receive an invocation of the *Federation Saved †* service. If a federate that received an invocation of the *Initiate Federate Save †* service resigns from the federation execution before the *Federation Saved †* service for that save is invoked, this resignation shall be considered a failure of the federation save, and the *Federation Saved †* service shall be invoked with a save-success indicator of failure.

Supplied Arguments

- Federation save-success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has been informed of the success or failure of the federation save attempt.
- The federate may resume providing new information to the federation.

Exceptions

- Federate internal error

Related Services

- *Request Federation Save*
- *Initiate Federate Save †*
- *Federate Save Begun*
- *Federate Save Complete*

4.16 Request Federation Restore

The *Request Federation Restore* service shall direct the RTI to begin the federation execution restoration process. Federation restoration shall begin as soon after the validation of the *Request Federation Restore* service invocation as possible. A valid federation restoration request shall be indicated with the *Confirm Federation Restoration Request* † service.

Supplied Arguments

- Federation save label

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federation has a save with the specified label.
- The correct number of federates of the correct types that were joined to the federation execution when the save was accomplished are currently joined to the federation execution.
- All previous *Request Federation Restore* service invocations from the federate have been acknowledged with a corresponding *Confirm Federation Restoration Request* †.

Post-conditions

- The RTI has been notified of the request to restore a former federation execution state.

Exceptions

- The federate not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Confirm Federation Restoration Request* †
- *Request Federation Save*
- *Federation Restore Begun* †
- *Initiate Federate Restore* †
- *Federate Restore Complete*
- *Federation Restored* †

4.17 Confirm Federation Restoration Request †

The *Confirm Federation Restoration Request* † shall indicate to the federate the status of a requested federation restoration. This service shall be invoked in response to a *Register Federation Restore* service invocation. A positive request success indicator informs the federate that the RTI restoration state information has been located which corresponds to the indicated label and federation execution name, a census of joined federates matches in number and type the census of federates present when the save was taken, and no other federate is currently attempting to restore the federation. Should more than one federate attempt to restore the federation at a given time, one federate shall receive a positive indication through this service and all others shall receive a negative indication. A federation restoration attempt that ends with a negative request success indicator shall have no other effect on the federation execution.

Supplied Arguments

- Federation save label
- Request success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has requested a federation restore via the *Register Federation Restore* service

Post-conditions

- If the request success indicator is positive, restore in progress.
- If the request success indicator is positive, the federation has a saved state with the specified label.
- If the request success indicator is positive, the correct number of federates of the correct types that were joined to the federation execution when the save was accomplished are currently joined to the federation execution.
- If the request success indicator is negative, this service and the corresponding *Request Federation Restore* service invocation have no consequence

Exceptions

- Federate internal error.

Related Services

- *Request Federation Restore*

4.18 Federation Restore Begun †

The *Federation Restore Begun* † service shall inform the federate that a federation restoration is imminent. The federate shall stop providing new information to the federation immediately after receiving the *Federation Restore Begun* † service invocation. The federate may resume providing new information to the federation only after receiving the *Federation Restored* † service invocation.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has been instructed to stop providing new information to the federation.

Exceptions

- Federate internal error

Related Services

- *Request Federation Restore*
- *Initiate Federate Restore* †
- *Federate Restore Complete*
- *Federation Restored* †

4.19 Initiate Federate Restore †

The *Initiate Federate Restore* † service shall instruct the federate to return to a previously saved state. The federate shall select the appropriate restoration state information based on the name of the current federation execution, the supplied federation save label, and the supplied federate type. As a result of this service invocation, a federate's designator could change from the value supplied by the *Join Federation Execution* service.

Supplied Arguments

- Federation save label
- Federate designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has a save with the specified label.

Post-conditions

- The federate has been informed to begin restoring state.

Exceptions

- There is no federate save associated with the label.
- Could not initiate restore
- Federate internal error

Related Services

- *Request Federation Restore*
- *Federation Restore Begun* †
- *Federate Restore Complete*
- *Federation Restored* †

4.20 Federate Restore Complete

The *Federate Restore Complete* service shall notify the RTI that the federate has completed its restore attempt. If restore was successful, the federate shall be in the state that either it or some other federate of its type was in when the federation save associated with the label occurred, with the distinction that the federate shall now be waiting for an invocation of the *Federation Saved* † service.

Supplied Arguments

- Federate restore-success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate was directed to restore through invocation of the *Initiate Restore* service.
- If restore was successful, the federate is in a state identical to the state that either it or some other federate of its type was in when the federation save associated with the supplied label occurred, with the distinction that the federate is now waiting for an invocation of the *Federation Saved* † service. If restore was unsuccessful, the federate is in an undefined state.

Post-conditions

- The RTI has been informed of the status of the restore attempt.

Exceptions

- Unknown label
- Invalid restore-success indicator
- Restore not requested
- The federate is not a federation execution member.
- Save in progress
- RTI internal error

Related Services

- *Request Federation Restore*
- *Federation Restore Begun* †
- *Initiate Federate Restore* †
- *Federate Restore Complete*
- *Federation Restored* †

4.21 Federation Restored †

The *Federation Restored* † service shall inform the federate that the federation restore process is complete, and shall indicate whether it completed successfully or not. If the restore-success indicator argument indicates success, this means that all federates at which the *Federation Restore Begun* † service was invoked have invoked the *Federate Restore Complete* service with a restore-success indicator that indicated success. If the restore-success indicator argument indicates failure, this means that one or more federates at which the *Federation Restore Begun* † service was invoked have invoked the *Federate Restore Complete* service with a restore-success indicator that indicated failure, or that the RTI detected failure at one or more of these federates. All federates that received an invocation of the *Federation Restore Begun* † service shall receive an invocation of the *Federation Restored* † service. If a federate that received an invocation of the *Federation Restore Begun* † service resigns from the federation execution before the *Federation Restored* † service for that restore is invoked, this resignation shall be considered a failure of the federation restoration, and the *Federation Restored* † service shall be invoked with a restore-success indicator of failure.

Supplied Arguments

- Federation restore-success indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has a save with the specified label.

Post-conditions

- The federate has been informed regarding the success or failure of the restoration attempt.
- The federate may resume providing new information to the federation.

Exceptions

- Federate internal error

Related Services

- *Request Federation Restore*
- *Federation Restore Begun* †
- *Initiate Federate Restore* †
- *Federate Restore Complete*

5. Declaration management

5.1 Overview

Federates shall use declaration management services to declare their intention to generate and receive information. A federate shall invoke appropriate declaration management services before it can register or discover object instances, update or reflect instance attribute values, and send or receive interactions. Declaration management services, together with object management services, ownership management services, and the object and interaction class hierarchies defined in the Federation Execution Data (FED) shall determine the

- Object classes at which object instances may be registered,
- Object classes at which object instances are discovered,
- Instance attributes that are available to be updated and reflected,
- Interactions that may be sent,
- Interaction classes at which interactions are received, and the
- Parameters that are available to be sent and received.

The effects of declaration management services shall be independent of federation time.

5.1.1 Static properties of the FED

The following static properties of the FED shall establish vocabulary for subsequent declaration management discussion:

- a) Every class shall have at most one immediate super-class. A class shall not be a super-class of a class that is its super-class.
- b) Every object class shall have an associated set of class attributes declared in the FED.
- c) An *inherited attribute* of an object class is a class attribute that was declared in a super-class.
- d) The *available attributes* of an object class are the set of declared attributes of that object class in union with the set of inherited attributes of that object class.
- e) Every interaction class shall have an associated set of parameters declared in the FED.
- f) An *inherited parameter* of an interaction class is a parameter that was declared in a super-class.
- g) The *available parameters* of an interaction class are the set of declared parameters of that interaction class in union with the set of inherited parameters of that interaction class.
- h) For any service that takes an object class and a set of attribute designators as arguments, only the available attributes of that object class may be used in the set of attribute designators. Being an available attribute of an object class shall be a necessary, but not necessarily a sufficient, condition for an attribute to be used in the set of attribute designators for such a service.
- i) For any service that takes an object instance and a set of attribute designators as arguments, only the available attributes of that object instance's known class at the involved (invoking or invoked) federate may be used in the set of attribute designators. Being an available attribute of the object instance's known class shall be a necessary, but not necessarily a sufficient, condition for an attribute to be used in the set of attribute designators for such a service.

5.1.2 Definitions and constraints for object classes and class attributes

The following declaration management definitions and constraints shall pertain to object classes and class attributes as declared in the class hierarchy of the FED:

- a) An attribute may be used as an argument to *Subscribe Object Class Attributes* and *Publish Object Class* service invocations for a particular object class if and only if the attribute is an available attribute of that object class.
- b) From a federate's perspective, the *subscribed attributes of an object class* shall be the class attributes that were arguments to the most recent *Subscribe Object Class Attributes* service invocation by that federate for that object class, assuming the federate did not subsequently invoke the *Unsubscribe Object Class* service for that object class. If the federate did subsequently invoke the *Unsubscribe Object Class* service for that object class, if the federate has not invoked the *Subscribe Object Class Attributes* service for that object class, or if the most recent *Subscribe Object Class Attributes* service invocation by that federate for that object class had an empty set of class attributes as argument, there shall be no subscribed attributes of that class for that federate. (*Subscribe Object Class Attributes* and *Unsubscribe Object Class* service invocations for one object class shall have no effect on the subscribed attributes of any other object class.)
- c) If a class attribute is a subscribed attribute of an object class, the federate shall be subscribed to that class attribute either actively or passively, but not both.
- d) From a federate's perspective, the *published attributes of an object class* shall be the class attributes that were arguments to the most recent *Publish Object Class* service invocation by that federate for that object class, assuming the federate did not subsequently invoke the *Unpublish Object Class* service for that object class. If the federate did subsequently invoke the *Unpublish Object Class* service for that object class, if the federate has not invoked the *Publish Object Class* service for that object class, or if the most recent *Publish Object Class Attributes* service invocation by that federate for that object class had an empty set of class attributes as argument, there shall be no published attributes of that class for that federate. (*Publish Object Class* and *Unpublish Object Class* service invocations for one object class shall have no effect on the published attributes of any other object class.)
- e) If a federate takes action that results in a class attribute that was a published attribute of its class no longer being a published attribute of its class, the federate shall be said to have *stopped publishing* that class attribute at that class. There shall be two ways that a federate can stop publishing a class attribute at a specific class: by invoking the *Unpublish Object Class* service for that object class or by invoking the *Publish Object Class* service for that object class without that class attribute designator among the arguments. These two ways of stopping publication of a class attribute shall be depicted by the labels *Unpublish* and *Publish (-i)* on the transition from the Published to the Unpublished state in the Publication state diagram of the Class Attribute (i) state (Figure 6).
- f) From a federate's perspective, an object class shall be *subscribed* if and only if
 - It was an argument to a *Subscribe Object Class Attributes* service invocation by that federate,
 - A non-empty set of class attributes was used as an argument to the most recent *Subscribe Object Class Attributes* service invocation for that object class by that federate, and
 - The most recent *Subscribe Object Class Attributes* service invocation for that object class by that federate was not subsequently followed by an *Unsubscribe Object Class* service invocation for the object class.
- g) From a federate's perspective, an object class shall be *published* if and only if:
 - It was an argument to a *Publish Object Class* service invocation by that federate,
 - A non-empty set of class attributes was used as an argument to the most recent *Publish Object Class* service invocation for that object class by that federate, and
 - The most recent *Publish Object Class* service invocation for that object class by that federate was not subsequently followed by an *Unpublish Object Class* service invocation for that object class.

- h) Federates may invoke the *Register Object Instance* service only with a published object class as an argument.
- i) The *registered class* of an object instance shall be the object class that was an argument to the *Register Object Instance* service invocation for that object instance.
- j) Every object instance shall have one federation-wide registered class that cannot change.
- k) An object instance shall have a *candidate discovery class* at a federate if and only if the federate is subscribed to either the registered class of the object instance or to a super-class of the registered class of the object instance. If an object instance has a *candidate discovery class* at a federate, the candidate discovery class of the object instance shall be the registered class of the object instance, if subscribed. Otherwise, it shall be the closest super-class of the registered class to which the federate is subscribed.
- l) If the *Discover Object* † service is invoked at a federate, the object instance discovered as a result of this service invocation shall have a *discovered class* at that federate. The discovered class of the object instance shall be the object instance's candidate discovery class at the time of the *Discover Object* † service invocation. The discovered class of the object instance shall be a supplied parameter to the *Discover Object* † service invocation.
- m) An object instance may have at most one discovered class in each federate. This discovered class may vary from federate to federate. Once an object instance is discovered, its discovered class shall not change. If a federate invokes the *Local Delete Object Instance* service for a given object instance, that object instance can be rediscovered. It may be rediscovered at a different discovered class.
- n) If a federate has registered or discovered an object instance and it has not subsequently
 - invoked the *Local Delete Object Instance* service for that object instance,
 - invoked the *Delete Object Instance* service for that object instance, or
 - received an invocation of the *Remove Object Instance* † service for that object instance,
 the object instance shall be known to that federate, and that object instance has a *known class* at that federate. The known class of that object instance at that federate shall be the object instance's registered class if the federate knows about the object instance as a result of having registered it. The *known class* of that object instance at that federate shall be the object instance's discovered class if the federate knows about the object instance as a result of having discovered it.
- o) A federate may own and update only an instance attribute for which it is publishing the corresponding class attribute at the known class of the instance attribute.
- p) An update to an instance attribute by the federate that owns that instance attribute can be reflected only to other federates that are subscribed to the corresponding class attribute at the instance attribute's known class at the subscribing federate.

5.1.3 Definitions and constraints for interaction classes and parameters

The following declaration management definitions and constraints shall pertain to interaction classes and parameters as declared in the interaction class hierarchy of the FED:

- a) From a federate's perspective, an interaction class shall be *subscribed* if and only if it was an argument to a *Subscribe Interaction Class* service invocation by that federate that was not subsequently followed by an *Unsubscribe Interaction Class* service invocation for that interaction class.

- b) If an interaction class is subscribed, the federate shall be subscribed to that interaction class either actively or passively, but not both.
- c) From a federate's perspective, an interaction class shall be *published* if and only if it was an argument to a *Publish Interaction Class* service invocation by that federate that was not subsequently followed by an *Unpublish Interaction Class* service invocation for that interaction class.
- d) Federates may invoke the *Send Interaction* service only with a published interaction class as an argument.
- e) The *sent class* of an interaction shall be the interaction class that was an argument to the *Send Interaction* service invocation for that interaction.
- f) Every interaction shall have one federation-wide sent class.
- g) The *Receive Interaction* † service can be invoked at a federate only with a subscribed interaction class as an argument.
- h) If the *Receive Interaction* † service is invoked at a federate, the interaction received as a result of this service invocation shall have a *received class* at that federate. The *received class* of an interaction is the interaction class that is an argument to the *Receive Interaction* † service invocation. The received class shall be the interaction's sent class, if subscribed. Otherwise, the received class shall be the closest super-class of the sent class that is subscribed at the time of the *Receive Interaction* † service invocation.
- i) An interaction may have at most one received class in each federate. This received class may vary from federate to federate.
- j) Only the available parameters of an interaction class may be used in a *Send Interaction* service invocation with that interaction class as argument.
- k) The *sent parameters* of an interaction shall be the parameters that were arguments to the *Send Interaction* service invocation for that interaction.
- l) The *received parameters* of an interaction shall be the parameters that were arguments to the *Receive Interaction* † service invocation for that interaction.
- m) The received parameters of an interaction shall be the subset of the sent parameters that are available parameters for the interaction's received class.
- n) The received parameters for a given interaction may vary from federate to federate, depending on the received class of the interaction.

When an object instance's discovered class is a super-class of its registered class, the object instance shall be said to have been *promoted* from the registered class to the discovered class. Similarly, when an interaction's received class is a super-class of its sent class, the interaction shall be said to have been promoted from the sent class to the received class. Promotion is important for protecting federate code from new subclasses added to the FED. As the FED is expanded to include new object and interaction classes, promotion ensures that existing federate code need not change to work with the expanded FED.

The following figures depict formal representations of the state of an arbitrary object class, an arbitrary class attribute, and an arbitrary interaction class. Figure 5 depicts the state of an arbitrary object class and it deals with object classes at two levels. First, it establishes that each class attribute of the object class has some state worth modeling. Second, it establishes that there are an arbitrary number of instances of each

object class. Further, it defines what conditions allow an object instance to be known by a federate as an instance of that object class.

Conceptually, the state of an object class shall comprise the state of the class attributes of that object class and of the object instances of that object class. The state of an object instance shall further comprise the state of the instance attributes of that object instance. There shall be a correspondence between the instance attributes and their corresponding class attributes. This correspondence shall be modeled via the index to each attribute. A reference within instance attribute (i) to something modeled at the class attribute (i) level shall mean that the i's are the same and thus the corresponding class attribute is being referenced.

Each object class shall have a fixed number of available class attributes as defined in the FED. The number of object instances of a given class, however, shall be arbitrary.

An object instance of an object class shall become known by the registering federate when the object instance is registered. It may become known by other federates in the federation execution. If it becomes known by other federates in the federation execution, it shall become known by them as a result of being discovered.

PLACE

Figure 5—Object class (i)

HERE

Figure 6 depicts the state of an arbitrary class attribute and shows the properties that may be controlled by a federate at the class attribute level. Specifically, a federate can publish or subscribe to class attributes. While the *Publish Object Class* and *Subscribe Object Class Attributes* service invocations can take sets of class attributes as an argument, Figure 6 depicts only a single class attribute. So, for example, “*Publish (i)*” shall mean that the *i*th class attribute was an element of the set used as an argument to the *Publish Object Class* service. A “*Publish (-i)*” shall mean that the *Publish Object Class* service was invoked, but that the *i*th class attribute was not an element of the set used as an argument to the service.

The federate may also direct the RTI via the *Enable/Disable Class Relevance Advisory Switch* services to indicate that the federate does or does not want the RTI to use the *Start Registration For Object Class †* and *Stop Registration For Object Class †* services to inform the federate when registration of new object instances are relevant to the other federates in the federation execution.

PLACE

Figure 6—Class attribute (i)

HERE

Figure 7 depicts the state of an arbitrary interaction class and shows the properties relating to interaction classes that may be controlled by a federate. Specifically, a federate can publish or subscribe to interaction classes.

The federate may also direct the RTI via the *Enable/Disable Interaction Relevance Advisory Switch* services to indicate that the federate does or does not want the RTI to use the *Turn Interactions On* † and *Turn Interactions Off* † services to inform the federate when interactions of a given class are relevant to the other federates in the federation execution.

PLACE

Figure 7—Interaction class (i)

HERE

5.2 Publish Object Class

The information conveyed by the federate via the *Publish Object Class* service shall be used in multiple ways. First, it shall indicate an object class of which the federate may subsequently register object instances. Second, it shall indicate the class attributes of the object class for which the federate is capable of owning the corresponding instance attributes of object instances whose known class is that class. Only the federate that owns an instance attribute can provide values for that instance attribute to the federation. The federate may become the owner of an instance attribute and thereby be capable of updating its value in two ways:

- By registering an object instance of a published class. Upon registration of an object instance, the registering federate shall become the owner of all instance attributes of that object instance for which the federate is publishing the corresponding class attributes at the registered class of the object instance.
- By using ownership management services to acquire instance attributes of object instances. The federate may acquire only those instance attributes of object instances for which the federate is publishing the corresponding class attributes at the known class of the object instance.

Each use of this service shall replace all information specified to the RTI in previous service invocations for the same object class. A class attribute that appears in this service invocation that also appeared in the previous service invocation for the same object class shall continue to be a published attribute of the specified object class. A class attribute that appears in this service invocation that did not appear in the previous service invocation for the same object class shall begin to be a published attribute of the specified class. A class attribute that does not appear in this service invocation but that did appear in the previous service invocation for the same object class shall stop being a published attribute of the specified class.

Invoking this service with an empty set of class attributes shall be equivalent to invoking the *Unpublish Object Class* service with the specified object class.

Supplied Arguments

- Object class designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified object class is defined in the FED.
- The specified class attributes are available attributes of the specified object class.
- If this service has been previously invoked for the same object class, then for each class attribute that was specified in the previous service invocation for this object class that was not specified in the current service invocation for this object class, there are no federate-owned corresponding instance attributes
 - that are part of an object instance whose known class is the specified class, and
 - for which the federate has either
 - a) invoked the Attribute Ownership Acquisition service, but has not yet received an invocation of either the *Confirm Attribute Ownership Acquisition Cancellation* † service or the *Attribute Ownership Acquisition Notification* † service, or

- b) invoked the *Attribute Ownership Acquisition If Available* service, but has not yet received an invocation of the *Attribute Ownership Unavailable* † service, received an invocation of the *Attribute Ownership Acquisition Notification* † service, or invoked the *Attribute Ownership Acquisition* service (after which condition 1 (above) applies.

Post-conditions

- The federate may now register object instances of the specified class.
- If the federate registers an object instance of the specified class, it shall own and may therefore update the instance attributes of that object instance that correspond to the specified class attributes.
- The specified class attributes shall now be published attributes of the specified object class. If there was a previous *Publish Object Class* service invocation for the specified object class by this federate, then for each class attribute that was specified in the previous service invocation that is not specified in the current service invocation (if any), the class attribute shall no longer be a published attribute of the specified object class.
- all corresponding instance attributes of object instances whose known class is the specified object class that were owned by the federate shall be unowned.

Exceptions

- The object class is not defined in the FED.
- The specified class attributes are not available attributes of the specified object class.
- Cannot Unpublish due to pending attempt to acquire instance attribute ownership.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Unpublish Object Class*
- *Subscribe Object Class Attributes*
- *Register Object Instance*
- *Attribute Ownership Divestiture Notification* †
- *Attribute Ownership Acquisition*
- *Attribute Ownership Acquisition If Available*

5.3 Unpublish Object Class

The *Unpublish Object Class* service shall inform the RTI that the federate will no longer register object instances of the specified object class. The federate shall lose ownership of all owned instance attributes of object instances whose known class is the specified object class, which means that the federate can no longer update any instance attribute values of object instances whose known class is the specified object class.

Supplied Arguments

- Object class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The federate is publishing the object class.
- For each class attribute that was specified in the most recent *Publish Object Class* service invocation for this object class, there are no federate-owned corresponding instance attributes
 - that are part of an object instance whose known class is the specified class, and
 - for which the federate has either
 - a) invoked the *Attribute Ownership Acquisition* service, but has not yet received an invocation of either the *Confirm Attribute Ownership Acquisition Cancellation* † service or the *Attribute Ownership Acquisition Notification* † service, or
 - b) invoked the *Attribute Ownership Acquisition If Available* service, but has not yet received an invocation of the *Attribute Ownership Unavailable* † service, received an invocation of the *Attribute Ownership Acquisition Notification* † service, or invoked the *Attribute Ownership Acquisition* service [after which condition (a) applies].

Post-conditions

- The federate may not register object instances of the specified object class.
- The federate shall no longer own any instance attributes of object instances whose known class is the specified object class.

Exceptions

- The object class is not defined in the FED.
- The federate is not publishing the object class.
- Cannot unpublish due to pending attempt to acquire instance attribute ownership.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Publish Object Class*
- *Attribute Ownership Divestiture Notification* †
- *Attribute Ownership Acquisition*

— *Attribute Ownership Acquisition If Available*

5.4 Publish Interaction Class

The *Publish Interaction Class* service shall inform the RTI which classes of interactions the federate will send to the federation execution.

Supplied Arguments

- Interaction class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is specified in the FED.

Post-conditions

- The federate may now send interactions of the specified class.

Exceptions

- The interaction class is not defined in the FED.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Unpublish Interaction Class*
- *Subscribe Interaction Class*
- *Send Interaction*

5.5 Unpublish Interaction Class

The *Unpublish Interaction Class* service shall inform the RTI that the federate will no longer send interactions of the specified class.

Supplied Arguments

- Interaction class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is specified in the FED.
- The federate is publishing the interaction class.

Post-conditions

- The federate may not send interactions of the specified interaction class.

Exceptions

- The interaction class is not defined in the FED.
- The federate is not publishing the interaction class.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Publish Interaction Class*

5.6 Subscribe Object Class Attributes

The *Subscribe Object Class Attributes* service shall specify an object class at which the RTI can notify the federate of discovery of object instances. When subscribing to an object class, the federate may also provide a set of class attributes. The values of only the instance attributes that correspond to the specified class attributes, for all object instances discovered as a result of this service invocation, shall be provided to the federate from the RTI (via the *Reflect Attribute Values* † service). The set of class attributes provided shall be a subset of the available attributes of the specified object class.

A federate can only discover an object as being of a class to which the federate is subscribed.

If a federate subscribes to multiple locations in an object class inheritance tree, each relevant object registration can result in at most one object discovery by the subscribing federate. The discovered class shall be the registered class, if subscribed by the discovering federate. Otherwise, the discovered class shall be the closest super-class of the registered class subscribed by the discovering federate.

Each use of this service shall replace all information specified to the RTI in any previous service invocations for the same object class.

Invoking this service with an empty set of class attributes shall be equivalent to invoking the *Unsubscribe Object Class* service with the specified object class.

If the optional passive subscription indicator indicates that this is a passive subscription,

- a) the invocation of this service shall not cause the *Start Registration For Object Class* † service to be invoked at any other federate and
- b) if this invocation replaces a previous subscription that was active rather than passive, invocation of this service may cause the *Stop Registration for Object Class* † service to be invoked at one or more other federates.

If the optional passive subscription indicator is not present or indicates that this is an active subscription, the invocation of this service may cause the *Start Registration For Object Class* service to be invoked at one or more other federates.

Supplied Arguments

- Object class designator
- Set of attribute designators
- Optional passive subscription indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified object class is defined in the FED.
- The specified class attributes are available attributes of the specified object class.

Post-conditions

- The RTI has been informed of the federate's requested subscription.

Exceptions

- The object class is not defined in the FED.
- The specified class attributes are not available attributes of the specified object class.
- Invalid passive subscription indicator.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Unsubscribe Object Class Attributes*
- *Publish Object Class*
- *Discover Object*
- *Attributes In Scope †*
- *Reflect Attribute Values †*
- *Start Registration For Object Class †*
- *Stop Registration for Object Class †*

5.7 Unsubscribe Object Class

The *Unsubscribe Object Class* service shall inform the RTI that it is to stop notifying the federate of object instance discovery at the specified object class. All in-scope instance attributes of known object instances whose known class is the specified object class shall go out of scope.

Supplied Arguments

- Object class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The federate is subscribed to the object class.

Post-conditions

- The federate shall receive no subsequent *Discover Object* service invocations for the specified object class.
- The federate shall receive no subsequent *Reflect Attribute Values* † service invocations for any instance attributes of object instances whose discovered class is the specified object class.

Exceptions

- The object class is not defined in the FED.
- The federate is not subscribed to the object class.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Subscribe Object Class Attributes*
- *Attributes Out Of Scope* †
- *Remove Object* †

5.8 Subscribe Interaction Class

Specifies an interaction class for which the RTI should notify the federate of sent interactions by invoking the *Receive Interaction* \dagger service at the federate.

When an interaction is received by a federate, the received class of the interaction shall be the interaction's sent class, if subscribed. Otherwise, the received class is the closest super-class of the sent class that is subscribed at the time the interaction is received. Only the parameters from the interaction's received class and its super-classes will be received.

If a federate subscribes to multiple locations in an interaction class inheritance tree, each relevant interaction sent can result in at most one received interaction in the subscribing federate.

If the optional passive subscription indicator indicates that this is a passive subscription,

- a) the invocation of this service shall not cause the *Turn Interactions On* \dagger service to be invoked at any other federate and
- b) if this invocation replaces a previous subscription that was active rather than passive, invocation of this service may cause the *Turn Interactions Off* \dagger service to be invoked at one or more other federates.

If the optional passive subscription indicator is not present or indicates that this is an active subscription, the invocation of this service may cause the *Turn Interactions On* \dagger service to be invoked at one or more other federates.

Supplied Arguments

- Interaction class designator
- Optional passive subscription indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.

Post-conditions

- The RTI will deliver interactions of the specified interaction class to the federate.

Exceptions

- The interaction class is not defined in the FED.
- Invalid passive subscription designator.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Unsubscribe Interaction Class*
- *Publish Interaction Class*
- *Receive Interaction* \dagger

- *Turn Interactions On* †
- *Turn Interactions Off* †

5.9 Unsubscribe Interaction Class

The *Unsubscribe Interaction Class* service shall inform the RTI to no longer notify the federate of sent interactions of the specified interaction class.

Supplied Arguments

- Interaction class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The federate is subscribed to the interaction class.

Post-conditions

- The RTI shall not deliver interactions of the specified interaction class to the federate.

Exceptions

- The interaction class is not defined in the FED.
- The federate is not subscribed to the interaction class.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Subscribe Interaction Class*

5.10 Start Registration For Object Class †

The *Start Registration For Object Class †* service shall notify the federate that registration of new object instances of the specified object class is advised because at least one of the class attributes that the federate is publishing at this object class is actively subscribed to at the specified object class or at a super-class of the specified object class by at least one other federate in the federation execution. The federate should commence with registration of object instances of the specified class. Generation of the *Start Registration For Object Class †* service advisory can be controlled using the *Enable/Disable Class Relevance Advisory Switch* services (Figure 5).

Supplied Arguments

- Object class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- At least one of the class attributes that the federate is publishing at the specified object class is actively subscribed to at the specified object class or at a super-class of the specified object class by at least one other federate in the federation execution.

Post-conditions

- The federate has been notified of the requirement to begin registering object instances of the specified object class.

Exceptions

- The object class is not published.
- Federate internal error

Related Services

- *Stop Registration For Object Class †*
- *Publish Object Class*
- *Subscribe Object Class Attributes*
- *Enable Class Relevance Advisory Switch*
- *Disable Class Relevance Advisory Switch*

5.11 Stop Registration For Object Class †

The *Stop Registration For Object Class †* service shall notify the federate that registration of new object instances of the specified object class is not advised because none of the class attributes that the federate is publishing at this object class is actively subscribed to at the specified object class or at a super-class of the specified object class by any other federate in the federation execution. The federate should stop registration of new object instances of the specified class. Generation of the *Stop Registration For Object Class †* service advisory can be controlled using the *Enable/Disable Class Relevance Advisory Switch* services (Figure 5).

Supplied Arguments

- Object class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- None of the class attributes that the federate is publishing at this object class is actively subscribed to at the specified object class or at a super-class of the specified object class by any other federate in the federation execution.

Post-conditions

- The federate has been notified of the requirement to stop registration of object instances of the specified object class.

Exceptions

- The object class is not published.
- Federate internal error

Related Services

- *Start Registration For Object Class †*
- *Publish Object Class*
- *Subscribe Object Class Attributes*
- *Unsubscribe Object Class Attributes*
- *Enable Class Relevance Advisory Switch*
- *Disable Class Relevance Advisory Switch*

5.12 Turn Interactions On †

The *Turn Interactions On †* service shall notify the federate that the specified class of interactions is relevant because it or a super-class is actively subscribed to by at least one other federate in the federation execution. The federate should commence with the federation-agreed-upon scheme for sending interactions of the specified class. Generation of the *Turn Interactions On †* service advisory can be controlled using the *Enable/Disable Interaction Relevance Advisory Switch* services (Figure 7).

Supplied Arguments

- Interaction class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate is publishing the interaction class.
- Some other federate is actively subscribed to the interaction class or to a super-class of the interaction class.

Post-conditions

- The federate has been notified that some other federate in the federation execution is subscribed to the interaction class.

Exceptions

- The interaction class is not published.
- Federate internal error

Related Services

- *Turn Interactions Off †*
- *Publish Interaction Class*
- *Subscribe Interaction Class*
- *Send Interaction*
- *Enable Interaction Relevance Advisory Switch*
- *Disable Interaction Relevance Advisory Switch*

5.13 Turn Interactions Off †

The *Turn Interactions Off †* service shall indicate to the federate that the specified class of interactions is not relevant because it or a super-class is not actively subscribed to by any other federate in the federation execution. Generation of the *Turn Interactions Off †* service advisory can be controlled using the *Enable/Disable Interaction Relevance Advisory Switch* services (Figure 7).

Supplied Arguments

- Interaction class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate is publishing the interaction class.
- No other federate is actively subscribed to the interaction class or to a super-class of the interaction class.

Post-conditions

- The federate has been notified that no other federate in the federation execution is subscribed to the interaction class.

Exceptions

- The interaction class is not published.
- Federate internal error

Related Services

- *Turn Interactions On †*
- *Publish Interaction Class*
- *Subscribe Interaction Class*
- *Unsubscribe Interaction Class*
- *Enable Interaction Relevance Advisory Switch*
- *Disable Interaction Relevance Advisory Switch*

6. Object management

6.1 Overview

This group of RTI services shall deal with the registration, modification, and deletion of object instances and the sending and receipt of interactions.

Object instance discovery is a prime concept in this service group. Object instance *O* shall be a candidate discovery class at federate *F* if federate *F* is subscribed to either the registered class of *O* or to a superclass of the registered class of *O*. If an object instance has a candidate discovery class at a federate, the candidate discovery class of the object instance at that federate shall be the object instance's registered class, if subscribed to by the federate. Otherwise, the candidate discovery class of the object instance shall be the closest superclass of the object instances's registered class to which the federate is subscribed.

A federate discovers an object instance via the *Discover Object Instance* † service. This service shall be invoked at a federate *F* for object instance *O* when

- a) *F* has not yet discovered *O*, and
- b) there is an instance attribute *i* of *O* that has a corresponding class attribute *i'*, and
 - Another federate (not *F*) owns *i*, and
 - *F* is subscribed to *i'* at *O*'s candidate discovery class

and, if data distribution management is considered, then

- c) the region with which the instance attribute *i* is associated at the owning federate for update purposes intersects with the region with which *F* has associated the instance attribute *i*'s corresponding class attribute *i'* for subscription purposes at the candidate discovery class.

When the *Discover Object Instance* † service is invoked, the class that is an argument to this service invocation shall be called the *discovered class* of the object instance. At the moment of discovery, the discovered class shall be the same as the candidate discovery class. At this moment of discovery, the discovered class shall be the registered class of the discovered object instance, if subscribed. If not, it shall be the closest superclass of the registered class to which *F* is subscribed. Subsequent to discovery, the discovered class cannot change. The candidate discovery class may change. As long as an object instance remains known, however, its candidate discovery class is not of interest.

When the *Discover Object Instance* † service is invoked, there shall be an instance attribute that is part of the newly discovered object instance that immediately comes into scope at the discovering federate, both when data distribution management is used and when it isn't used. An instance attribute of an object instance shall be in scope for federate *F* if

- a) The object instance is known to the federate,
- b) The instance attribute's corresponding class attribute is subscribed at the known class of the instance attribute,
- c) The instance attribute is owned by another federate,

and if data distribution management is used by both the federate that owns the instance attribute and federate *F*, then

- d) The region to which the instance attribute is associated at the owning federate for update purposes intersects with the region to which the instance attribute's corresponding class attribute is associated at the subscribing federate at the known class of the instance attribute.

A federate may also direct the RTI, via the *Enable/Disable Attribute Relevance Advisory Switch* services, to indicate that the federate does or does not want the RTI to use the *Turn Updates On For Object Instance \dagger* and *Turn Updates Off For Object Instance \dagger* services to inform the federate when updates to instance attributes corresponding to this class attribute are relevant to the other federates in the federation execution.

The following statecharts depict formal representations of the state of an arbitrary object instance, an arbitrary instance attribute, and the implications of ownership of an arbitrary instance attribute.

PLACE

Figure 8—Object instance (i) known

HERE

PLACE

Figure 9—Instance attribute (i)

HERE

PLACE

Figure 10—Implications of ownership of instance attribute (i)

HERE

6.2 Register Object Instance

The RTI shall create a unique object instance designator and shall link it with an instance of the supplied object class. All instance attributes of the object instance for which the corresponding class attributes are currently published by the registering federate shall be set as owned by the registering federate.

If the optional object instance name argument is supplied, that name shall be unique and shall be associated with the object instance. If the optional object instance name argument is not supplied, the RTI shall create one when needed (*Get Object Instance Name* service).

Supplied Arguments

- Object class designator
- Optional object instance name

Returned Arguments

- Object instance designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The federate is publishing the object class.
- If the optional object instance name argument is supplied, that name is unique.

Post-conditions

- The returned object instance designator is associated with the object instance.
- The federate owns the instance attributes that correspond to the currently published class attributes for the specified object class.
- If the optional object instance name argument is supplied, that name is associated with the object instance.

Exceptions

- The object class is not defined in FED.
- The federate is not publishing the specified object class.
- The object instance name is not unique.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Publish Object Class*
- *Discover Object Instance* †
- *Attribute Ownership Divestiture Notification* †
- *Get Object Instance Name*
- *Get Object Instance Handle*

6.3 Discover Object Instance †

The *Discover Object Instance* † service shall inform the federate to discover an object instance. An object instance shall be discovered when the instance has been registered by another federate or as the result of a *Local Delete Object Instance* service invocation.

Supplied Arguments

- Object instance designator
- Object class designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is published by some federate.
- The federate is subscribed to the object class.
- The instance of the class has been registered by another federate.
- The federate does not know about the object instance with the specified designator.

Post-conditions

- The object instance is known to the federate.

Exceptions

- The federate could not discover the object instance.
- The object class is not known.
- Federate internal error

Related Services

- *Register Object Instance*
- *Subscribe Object Class*
- *Local Delete Object Instance*

6.4 Update Attribute Values

The *Update Attribute Values* service shall provide current values to the federation for instance attributes owned by the federate. The federate shall supply changed instance attribute values as specified in the FED. This service, coupled with the *Reflect Attribute Values* † service, shall form the primary data exchange mechanism supported by the RTI. The service shall return a federation-unique event retraction designator. An event retraction designator shall be returned only if the federation time argument is supplied.

Supplied Arguments

- Object instance designator
- Set of attribute designator and value pairs
- User-supplied tag
- Optional federation time

Returned Arguments

- Optional event retraction designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate owns the instance attributes for which values are provided.
- The attributes are defined in the FED.
- An object instance with the specified designator exists.

Post-conditions

- The RTI will distribute the new instance attribute values to subscribing federates.

Exceptions

- The object instance is not known.
- The specified class attributes are not available attributes of the instance object class.
- The federate does not own the specified instance attributes.
- The federation time is invalid (if optional time argument is supplied).
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Reflect Attribute Values* †

6.5 Reflect Attribute Values †

The *Reflect Attribute Values* † service shall provide the federate with new values for the specified instance attributes. This service, coupled with the *Update Attribute Values* service, shall form the primary data exchange mechanism supported by the RTI.

All the instance attribute/value pairs in an *Update Attribute Values* service invocation for instance attributes that have identical transportation and message-ordering types shall be in one corresponding *Reflect Attribute Values* † service invocation. This implies that one *Update Attribute Values* invocation could result in multiple *Reflect Attribute Values* † invocations in a subscribing federate. The federation time and event retraction designator arguments shall be supplied together or not at all.

Supplied Arguments

- Object instance designator
- Set of attribute designator and value pairs
- User-supplied tag
- Optional federation time
- Optional event retraction designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate is subscribed to the attributes.
- The federate does not own the instance attributes.

Post-conditions

- The new instance attribute values have been supplied to the federate.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The instance attribute is owned by the federate.
- The federation time is invalid (if optional time argument is supplied).
- Federate internal error

Related Services

- *Update Attribute Values*
- *Time Advance Request*
- *Next Event Request*
- *Time Advance Grant* †

6.6 Send Interaction

The *Send Interaction* service shall send an interaction into the federation. The interaction parameters may be those in the specified class and all super-classes, as defined in the FED. The service shall return a federation-unique event retraction designator. An event retraction designator shall be returned only if the federation time argument is supplied.

Supplied Arguments

- Interaction class designator
- Set of interaction parameter designator and value pairs
- User-supplied tag
- Optional federation time

Returned Arguments

- Optional event retraction designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate is publishing the interaction class.
- The interaction class is defined in the FED.
- The parameters are defined in the FED.

Post-conditions

- The RTI has received the interaction.

Exceptions

- The federate is not publishing the specified interaction class.
- The interaction class is not defined in FED.
- The interaction parameter is not defined in FED.
- The federation time is invalid (if optional time argument is supplied).
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Next Event Request*
- *Time Advance Grant* †
- *Receive Interaction* †
- *Publish Interaction Class*
- *Retract*

6.7 Receive Interaction †

The *Receive Interaction* † service shall provide the federate with a sent interaction. The federation time and event retraction designator arguments shall be supplied together or not at all.

Supplied Arguments

- Interaction class designator
- Set of interaction parameter designator and value pairs
- User-supplied tag
- Optional federation time
- Optional event retraction designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate is subscribed to the interaction class.

Post-conditions

- The federate has received the interaction.

Exceptions

- The interaction class is not known.
- The interaction parameter is not known.
- The federation time is invalid (if optional time argument is supplied).
- Federate internal error

Related Services

- *Retract*
- *Send Interaction*
- *Subscribe Interaction Class*

6.8 Delete Object Instance

The *Delete Object Instance* service shall inform the federation that an object instance with the specified designator, owned by the federate, is to be removed from the federation execution. Once the object instance is removed from the federation execution, the designator shall not be reused and all federates which owned attributes of the object instance no longer own those attributes. The RTI shall use the *Remove Object* service to inform the reflecting federates that the object instance has been deleted. The invoking federate shall own the *privilegeToDeleteObject* attribute of the specified object instance. The preferred order type of the sent message representing a *Delete Object Instance* service invocation shall be based on the preferred order type of the *privilegeToDeleteObject* attribute of the specified object instance. (see clause 8 for more detail). An event retraction designator shall be returned only if the federation time argument is supplied.

Supplied Arguments

- Object instance designator
- User-supplied tag
- Optional federation time

Returned Arguments

- Optional event retraction designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate has the privilege to delete the object instance (it owns the *privilegeToDeleteObject* instance attribute).

Post-conditions

- The invoking federate may no longer update any previously owned attributes of the specified object instance.
- The object instance does not exist in the federation execution.

Exceptions

- The federate does not own the delete privilege.
- The object instance is not known.
- The federation time is invalid (if optional time argument is supplied).
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Remove Object Instance* †
- *Attribute Ownership Divestiture Notification* †

6.9 Remove Object Instance †

The *Remove Object Instance* † service shall inform the federate that an object instance has been deleted from the federation execution. The federation time and event retraction designator arguments shall be supplied together or not at all.

Supplied Arguments

- Object instance designator
- User-supplied tag
- Optional federation time
- Optional event retraction designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.

Post-conditions

- The federate has been notified to remove the object instance and may not update any previously owned attributes of the object instance.

Exceptions

- The object instance is not known.
- The federation time is invalid (if optional time argument is supplied).
- Federate internal error

Related Services

- *Delete Object Instance*

6.10 Local Delete Object Instance

The *Local Delete Object Instance* service shall inform the RTI that it shall treat the specified object instance as if the RTI had never notified the invoking federate to discover the object instance. The object instance shall not be removed from the federation execution. The federate does not need to own the *privilegeToDeleteObject* instance attribute for the object instance.

Supplied Arguments

- Object instance designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate owns no attributes of the specified object instance.

Post-conditions

- The object instance does not exist with respect to the invoking federate.
- The object instance may be rediscovered by the invoking federate, at a possibly different class than previously discovered.

Exceptions

- The object instance is not known.
- The federate owns instance attributes.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Attribute Ownership Divestiture Notification* †

6.11 Change Attribute Transportation Type

The transportation type for each attribute of an object instance shall be initialized from the object class description in the FED. A federate may choose to change the transportation type during execution. Invoking the *Change Attribute Transportation Type* service shall change the transportation type for all future *Update Attribute Values* service invocations for the specified attributes of the specified object instance only for the invoking federate.

Supplied Arguments

- Object instance designator
- Set of attribute designators
- Transportation type

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The specified class attributes are available attributes of the known class of the specified object instance designator.
- The federate owns the instance attributes.

Post-conditions

- The transportation type is changed for the specified instance attributes.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate does not own the specified instance attributes.
- The transportation type is invalid.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Update Attribute Values*
- *Change Attribute Order Type*

6.12 Change Interaction Transportation Type

The transportation type for each interaction shall be initialized from the interaction class description in the FED. A federate may choose to change the transportation type during execution. Invoking the *Change Interaction Transportation Type* service shall change the transportation type for all future *Send Interaction* and *Send Interaction with Region* service invocations for the specified interaction class for the invoking federate only.

Supplied Arguments

- Interaction class designator
- Transportation type

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The federate is publishing the interaction class.

Post-conditions

- The transportation type is changed for the specified interaction class.

Exceptions

- The interaction class is not defined in FED.
- The federate is not publishing the interaction class.
- The transportation type is invalid.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Send Interaction*
- *Change Interaction Order Type*

6.13 Attributes In Scope †

The *Attributes In Scope* † service shall notify the federate that the specified attributes for the object instance are in scope for the federate. Subsequent to this service invocation, the RTI may issue *Reflect Attribute Values* † service invocations for any of the set of attributes for the object instance. Generation of the *Attributes In Scope* † service advisory can be controlled using the *Enable/Disable Attribute Scope Advisory Switch* services.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate is subscribed to the class attributes.
- The federate does not own the instance attributes.
- If there are regions involved, they overlap (see clause 9).

Post-conditions

- The RTI is allowed to issue *Reflect Attribute Values* † service invocations for any of the set of attributes of the object instance.
- The federate is ready to accept *Reflect Attribute Values* † service invocations for any of the set of attributes of the object instance.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- Federate internal error

Related Services

- *Attributes Out Of Scope* †
- *Reflect Attribute Values* †
- *Enable Attribute Scope Advisory Switch*
- *Disable Attribute Scope Advisory Switch*

6.14 Attributes Out Of Scope †

The *Attributes Out Of Scope* † service shall notify the federate that the specified attributes of the object instance are out of scope for the federate. The RTI shall guarantee not to issue any subsequent *Reflect Attribute Values* † service invocations for any of the set of attributes for the object instance. Generation of the *Attributes Out Of Scope* † service advisory can be controlled using the *Enable/Disable Attribute Scope Advisory Switch* services.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- At least one of the following is not true:
 - The federate knows about the object instance with the specified designator.
 - The federate is subscribed to the class attributes.
 - The federate does not own the instance attributes.
 - If there are regions involved, they overlap (see clause 9).

Post-conditions

- The RTI guarantees not to issue *Reflect Attribute Values* † service invocations for any of the set of attributes of the object instance.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- Federate internal error

Related Services

- *Attributes In Scope* †
- *Reflect Attribute Values* †
- *Enable Attribute Scope Advisory Switch*
- *Disable Attribute Scope Advisory Switch*

6.15 Request Attribute Value Update

The *Request Attribute Value Update* service shall be used to stimulate the update of values of specified attributes. When this service is used, the RTI shall solicit the current values of the specified attributes from their owners using the *Provide Attribute Value Update* † service. When an object class is specified, the RTI shall solicit the specified attributes for all the object instances of that class. When an object instance designator is specified, the RTI shall solicit the specified attributes for the particular object instance. The federation time of any resulting *Reflect Attribute Values* † service invocations is determined by the updating federate.

Supplied Arguments

- Object instance designator or object class designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists (when first argument is an object instance designator).
- The specified class attributes are available attributes of the known class of the specified object instance designator (when first argument is an object instance designator).
- The specified object class is defined in the FED (when first argument is an object class).
- The specified class attributes are available attributes of the specified object class (when first argument is an object class).

Post-conditions

- The request for the updated attribute values has been received by the RTI.

Exceptions

- The object instance is invalid.
- The object class is not defined in FED.
- The class attribute is not available at the known class of the object instance.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Provide Attribute Value Update* †
- *Update Attribute Values*

6.16 Provide Attribute Value Update †

The *Provide Attribute Value Update* † service shall request the current values for attributes owned by the federate for a given object instance. The federate shall respond to the *Provide Attribute Value Update* † service with an invocation of the *Update Attribute Values* service to provide the requested instance attribute values to the federation.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate owns the specified instance attributes.

Post-conditions

- The federate has been notified to provide updates of the specified instance attribute values.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The instance attribute is not owned.
- Federate internal error

Related Services

- *Request Attribute Value Update*
- *Update Attribute Values*

6.17 Turn Updates On For Object Instance †

The *Turn Updates On For Object Instance †* service shall indicate to the federate that the values of the specified attributes of the specified object instance are required somewhere in the federation execution. The federate shall commence with the federation-agreed-upon update scheme for the specified instance attributes. Generation of the *Turn Updates On For Object Instance †* service advisory can be controlled using the *Enable/Disable Attribute Relevance Advisory Switch* services.

Supplied Arguments

- Object instance designator
- Set of attribute designators type

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate owns the instance attributes.
- The federate knows about the object instance with the specified designator.
- Some other federate in the execution is actively subscribed to the attributes of the object class.

Post-conditions

- The federate has been notified by another federate in the federation execution of the requirement for updates of the specified attributes of the specified object instance.

Exceptions

- The object instance is not known.
- The instance attribute is not owned.
- Federate internal error

Related Services

- *Turn Updates Off For Object Instance †*
- *Publish Object Class*
- *Subscribe Object Class Attributes With Region*
- *Update Attribute Values*
- *Enable Attribute Relevance Advisory Switch*
- *Disable Attribute Relevance Advisory Switch*

6.18 Turn Updates Off For Object Instance †

The *Turn Updates Off For Object Instance †* service shall indicate to the federate that the values of the specified attributes of the object instance are not required anywhere in the federation execution. Generation of the *Turn Updates Off For Object Instance †* service advisory can be controlled using the *Enable/Disable Attribute Relevance Advisory Switch* services.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate owns the specified instance attributes.
- The federate knows about the object instance with the specified designator.
- No other federate is actively subscribed to the attributes of the object class.

Post-conditions

- The federate has been notified by another federate in the federation execution that updates of the specified attributes of the specified object instance are not required.

Exceptions

- The object instance is not known.
- The attribute is not owned.
- Federate internal error

Related Services

- *Turn Updates On For Object Instance †*
- *Publish Object Class*
- *Subscribe Object Class Attributes With Region*
- *Unsubscribe Object Class Attributes With Region*
- *Update Attribute Values*
- *Enable Attribute Relevance Switch*
- *Disable Attribute Relevance Switch*

7. Ownership management

7.1 Overview

Ownership management shall be used by federates and the RTI to transfer ownership of instance attributes among federates. The ability to transfer ownership of instance attributes among federates shall be required to support the cooperative modeling of a given object instance across a federation. Only the federate that owns an instance attribute

- May invoke the *Update Attribute Values* service to provide a new value for that instance attribute,
- Can receive invocations of the *Provide Attribute Value Update †* service for that instance attribute, and
- Can receive invocations of the *Turn Updates On For Object Instance †* and *Turn Updates Off For Object Instance †* services pertaining to that instance attribute.

Figure 11 depicts the ways that ownership of a single instance attribute can be established from the viewpoint of a given federate. This diagram is complete insofar as all transitions shown represent legal operations, and transitions that are not shown represent illegal operations. Illegal operations shall generate exceptions if invoked.

PLACE

Figure 11—Establishing ownership of instance attribute (i)

HERE

An instance attribute shall not be owned by more than one federate at any given time, and an instance attribute may be unowned by all federates. From a given federate's perspective, every instance attribute shall be either owned or unowned. Hence, within the state machine depicted in Figure 11, the owned and unowned states are exclusive.

Upon registration of an object instance, the registering federate shall own all instance attributes of that object instance for which the federate is publishing the corresponding class attributes at the registered class of the object instance. All other instance attributes of that object instance shall be unowned by all federates. Upon discovery of an object instance, the discovering federate shall not own any instance attributes of that object instance. If a federate does not own an instance attribute, it shall not own that instance attribute until it has received an *Attribute Ownership Acquisition Notification* † (AOAN †) service invocation for it.

Within the owned state there shall be two parallel state machines for divestiture and release, meaning that an instance attribute is in both of these machines simultaneously. Each of these state machines shall have two exclusive states. An instance attribute that is owned is either in the process of being divested or not in the process of being divested. Simultaneously, a request to release it has either been received by its owning federate or not. Upon becoming owned, an instance attribute is initially not in the process of being divested and, simultaneously, no request to release it has yet been received. Because the divestiture and release state machines of Figure 11 operate in parallel, a federate may, for example, respond to a *Request Attribute Ownership Release* † service invocation with an *Unconditional Attribute Ownership Divestiture* or *Negotiated Attribute Ownership Divestiture* service invocation.

Ownership of an instance attribute can be transferred from one federate to another either by the owning federate requesting to divest itself of the instance attribute or by a non-owning federate requesting to acquire it. Whether an instance attribute changes ownership as a result of being divested by its owner or acquired by a non-owner, however, the instance attribute shall change ownership only as a result of explicit service invocations by the owning and acquiring federates. Ownership shall not be taken away from, nor shall it be given to, a federate without the federate's consent.

7.1.1 Ownership and publication

The ownership of an instance attribute is closely related to whether that instance attribute's corresponding class attribute is published at the known class of the instance attribute. This means that the ownership state machine in Figure 11, which operates in parallel with the publication state machine in Figure 6, also shares interdependencies with the publication state machine. A federate shall be publishing a class attribute at the known class of an object instance in order to own the corresponding instance attribute of that object instance. This means that

- A federate shall be publishing a class attribute at the known class of an object instance before it can become the owner of the corresponding instance attribute of that object instance. This interdependency between ownership and publication is expressed in Figure 11 by the Not Able to Acquire state, the [in "Unpublished (i)"] and [in "Published (i)"] transitions in the Unowned state, and the conditional transition into the Owned and Unowned states from the start state.
- if the federate that owns an instance attribute stops publishing the corresponding class attribute at the known class of the instance attribute, the instance attribute shall immediately become unowned. This interdependency between ownership and publication is expressed in Figure 11 by the transition from the Owned to the Unowned state that is labeled [in "Unpublished (i)"]. As depicted by the guard on the transition from the Published to the Unpublished state in the publication state machine shown in Figure 6, a federate shall not stop publication of a class attribute at a given class if there is an object instance that has that class as its known class and that has a corresponding instance attribute that is in either the Acquisition Pending or Willing to Acquire state at that federate. That is, a federate shall not stop publishing a class attribute at a given class if there is an object instance that has that class as its known class and that has a corresponding instance attribute for which the federate has
 - a) Invoked the *Attribute Ownership Acquisition* service, but has not yet received an invocation of either the *Confirm Attribute Ownership Acquisition Cancellation* † service or the *Attribute Ownership Acquisition Notification* † service, or
 - b) Invoked the *Attribute Ownership Acquisition If Available* service, but has not yet received an invocation of the *Attribute Ownership Unavailable* † service, received an invocation of the *Attribute Ownership Acquisition Notification* † service, or invoked the *Attribute Ownership Acquisition* service [after which condition (a) (above) applies].

7.1.2 Ownership transfer

An instance attribute that is successfully divested shall become unowned by the divesting federate. If an instance attribute is unowned, its corresponding class attribute at the known class of the instance attribute may be either published or unpublished. If the class attribute is published at that class, the federate shall be eligible to acquire the corresponding instance attribute and it may be offered ownership of that instance attribute by the RTI via the *Request Attribute Ownership Assumption* † service. There are five ways in which an owning federate may attempt to divest itself of an instance attribute and two ways in which a non-owning federate may attempt to acquire one.

7.1.2.1 Divestiture

The five actions that a federate could take to cause an instance attribute that it owns to become unowned are

- a) The federate can invoke the *Unconditional Attribute Ownership Divestiture* service, in which case the instance attribute shall immediately become unowned by that federate and, in fact, by all federates.

- b) The federate can invoke the *Negotiated Attribute Ownership Divestiture* service, which notifies the RTI that the federate wishes to divest itself of the instance attribute providing that the RTI can locate a federate that is willing to own the instance attribute. If any federates are in the process of trying to acquire the instance attribute, these federates are willing to own the instance attribute. The RTI can try to identify other federates that are willing to own the instance attribute by invoking the *Request Attribute Ownership Assumption* † service at all federates that are not in the process of trying to acquire the instance attribute, but that are publishing the instance attribute's corresponding class attribute at the known class of the instance attribute. If the RTI is able to locate a federate that is willing to acquire the instance attribute, the RTI shall notify the divesting federate that it no longer owns the instance attribute by invoking the *Attribute Ownership Divestiture Notification* † (AODN †) service at the divesting federate.
- c) The federate can invoke the *Attribute Ownership Release Response* service (in response to having received an invocation of the *Request Attribute Ownership Release* † service for the designated instance attribute). This service invocation shall have a return argument that the RTI shall use to indicate the set of instance attributes that have been successfully released. So, if the *Attribute Ownership Release Response* service returns with the designated instance attribute among the set of released instance attributes, the instance attribute shall be unowned. [In Figure 11, the transition from the owned to the unowned state via an *Attribute Ownership Release Response* service invocation is labeled *Release Response (ret: success)*]. This is a convenience notation indicating that the instance attribute in question is a member of the returned instance attribute set.
- d) The federate can stop publishing the instance attribute's corresponding class attribute at the known class of the instance attribute, which shall result in the instance attribute immediately becoming unowned by that federate and, in fact, by all federates.
- e) The federate can resign from the federation execution. When a federate successfully resigns from the federation execution, all of the instance attributes that are owned by that federate shall become unowned by that federate and, in fact, by all federates. This transition is not depicted in Figure 11 because it occurs at a federate, rather than an instance attribute, level of operation.

Of the five ways a federate may divest itself of an instance attribute, only the *Negotiated Attribute Ownership Divestiture* service may be canceled. A *Negotiated Attribute Ownership Divestiture* service invocation shall remain pending until either the instance attribute becomes unowned or the divesting federate cancels the divestiture request by invoking the *Cancel Negotiated Attribute Ownership Divestiture* service. Cancellation of the divestiture shall be guaranteed to be successful.

Of the five ways a federate may divest itself of an instance attribute, three ways (invocation of the *Unconditional Attribute Ownership Divestiture* service, a request to stop publication of the instance attribute's corresponding class attribute at the known class of the instance attribute, and invocation of the *Resign Federation Execution* service) shall result in the instance attribute becoming unowned by all federates. When either the *Negotiated Attribute Ownership Divestiture* or the *Attribute Ownership Release Response* service is used, the RTI shall guarantee that immediately after the owning federate loses ownership of the instance attribute, another federate shall be granted ownership of it. For purposes of determining an instance attribute's scope, the instance attribute may be considered to be continuously owned during its transfer of ownership from the divesting federate to the acquiring federate via either the *Negotiated Attribute Ownership Divestiture* or the *Attribute Ownership Release Response* service.

7.1.2.2 Acquisition

There shall be two ways for a federate that is publishing a class attribute at a given class to acquire a corresponding instance attribute of an object that has that class as its known class.

- a) The federate may invoke the *Attribute Ownership Acquisition* service, which shall inform the RTI that it shall invoke the *Request Attribute Ownership Release* † service at the federate that owns the designated instance attribute.
- b) The federate may invoke the *Attribute Ownership Acquisition If Available* service, which shall inform the RTI that it wants to acquire the designated instance attribute only if it is already unowned by all federates or if it is in the process of being divested by its owner.

The first method of acquisition can be thought of as an intrusive acquisition, because the RTI will notify the federate that owns the instance attribute that another federate wants to acquire it and request that the owning federate release the instance attribute for acquisition by the requesting federate. The second method of acquisition can be thought of as a non-intrusive acquisition because the RTI will not notify the owning federate of the request to acquire the instance attribute. The *Attribute Ownership Acquisition* service can also be thought of as taking precedence over the *Attribute Ownership Acquisition If Available* service. A federate that has invoked the *Attribute Ownership Acquisition* service and is in the Acquisition Pending state shall not invoke the *Attribute Ownership Acquisition If Available* service. If a federate that has invoked the *Attribute Ownership Acquisition If Available* service and is in the Willing to Acquire state invokes the *Attribute Ownership Acquisition* service, that federate shall enter the Acquisition Pending state.

An *Attribute Ownership Acquisition* service invocation may be explicitly canceled, but an *Attribute Ownership Acquisition If Available* service invocation shall not be explicitly cancelled. When a federate invokes the *Attribute Ownership Acquisition If Available* service, either the *Attribute Ownership Acquisition Notification* † service or the *Attribute Ownership Unavailable* † service shall be invoked at that federate in response. (If the instance attribute is unowned by all federates or in the process of being divested by its owner, the *Attribute Ownership Acquisition Notification* † service shall be invoked. Otherwise, the *Attribute Ownership Unavailable* † service shall be invoked.)

When a federate invokes the *Attribute Ownership Acquisition* service invocation, on the other hand, this request shall remain pending until either the instance attribute is acquired (as indicated by an invocation of the *Attribute Ownership Acquisition Notification* † service) or the federate successfully cancels the acquisition request. A federate may attempt to cancel the acquisition request by invoking the *Cancel Attribute Ownership Acquisition* service. The *Cancel Attribute Ownership Acquisition* service is not guaranteed to be successful. If it is successful, the RTI shall indicate this success to the canceling federate by invoking the *Confirm Attribute Ownership Acquisition Cancellation* † service. If it fails, the RTI shall indicate this failure to the canceling federate by invoking the *Attribute Ownership Acquisition Notification* † service, thereby granting ownership of the instance attribute to the federate.

An *Attribute Ownership Acquisition* service invocation shall override an *Attribute Ownership Acquisition If Available* service invocation. This means that a federate that has invoked the *Attribute Ownership Acquisition If Available* service may, before it receives an invocation of either the *Attribute Ownership Acquisition Notification* † service or the *Attribute Ownership Unavailable* † service, invoke the *Attribute Ownership Acquisition* service. In this case, the *Attribute Ownership Acquisition If Available* service request shall be implicitly canceled and the *Attribute Ownership Acquisition* service request shall remain pending until either the instance attribute is acquired or the federate successfully cancels the acquisition request. A federate that has invoked the *Attribute Ownership Acquisition* service, but has not yet received an invocation of either the *Attribute Ownership Acquisition Notification* † service or the *Confirm Attribute*

Ownership Acquisition Cancellation † service, shall not invoke the *Attribute Ownership Acquisition If Available* service.

7.1.3 PrivilegeToDeleteObject

All object classes shall have an available attribute called *privilegeToDeleteObject*. As with all other available attributes, a federate shall be publishing the *privilegeToDeleteObject* class attribute at the known class of an object instance to own the corresponding *privilegeToDeleteObject* instance attribute that is part of that object instance, and ownership of *privilegeToDeleteObject* instance attributes can be transferred among federates. Ownership management services for *privilegeToDeleteObject* instance attributes shall be the same as they are for all other instance attributes. The reason that a federate may want to own the *privilegeToDeleteObject* instance attribute, however, is different. Ownership of a typical instance attribute shall give a federate the privilege to provide new values for that instance attribute. Ownership of the *privilegeToDeleteObject* instance attribute of an object instance shall give the federate the additional right to delete that object instance from the federation execution.

7.1.4 User-supplied tags

Several of the ownership management services take a user-supplied tag as argument. These arguments shall be provided as a mechanism for conveying information between federates that could be used to implement priority or other schemes. While the content and use of these tags are outside of the scope of this specification, the RTI shall pass these user-supplied tags from federates that are trying to acquire an instance attribute to the federate that owns the instance attribute, and from the federate that is trying to divest itself of an instance attribute to the federates that are able to acquire the instance attribute. In particular,

- The user-supplied tag present in the *Negotiated Attribute Ownership Divestiture* service shall be present in any resulting *Request Attribute Ownership Assumption* † service invocations.
- The user-supplied tag present in the *Request Attribute Ownership Acquisition* service shall be present in any resulting *Request Attribute Ownership Release* † service invocations.

7.1.5 Sets of attribute designators

While many of the ownership management services take a set of instance attributes as an argument, the RTI treats ownership management operations on a per-instance-attribute basis. The fact that some ownership management service invocations take sets of instance attributes as an argument is a feature provided to federate designers for convenience. A single request with an instance attribute set as an argument can result in multiple responses pertaining to disjoint subsets of those instance attributes. For example, a single *Negotiated Attribute Ownership Divestiture* that has a set of instance attributes as an argument could result in multiple *Attribute Ownership Divestiture Notification* † service invocations. If one instance attribute in the set of instance attributes provided as an argument to an ownership management service invocation violates the preconditions of the service, an exception shall be generated and the entire service invocation shall fail.

7.2 Unconditional Attribute Ownership Divestiture

The *Unconditional Attribute Ownership Divestiture* service shall notify the RTI that the federate no longer wants to own the specified instance attributes of the specified object. This service shall immediately relieve the divesting federate of the ownership, causing the instance attribute(s) to go (possibly temporarily) into the unowned state, without regard to the existence of an accepting federate. Completion of the invocation of this service can be viewed as an implied invocation of the *Attribute Ownership Divestiture Notification* † service for all of the specified instance attributes.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate owns the specified instance attributes.

Post-conditions

- The federate no longer owns the specified instance attributes.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate does not own the instance attribute.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Attribute Ownership Divestiture Notification* †

7.3 Negotiated Attribute Ownership Divestiture

The *Negotiated Attribute Ownership Divestiture* service shall notify the RTI that the federate no longer wants to own the specified instance attributes of the specified object instance. Ownership shall be transferred only if some federate(s) accepts. The invoking federate shall continue its update responsibility for the specified instance attributes until it receives permission to stop via the *Attribute Ownership Divestiture Notification* † service. The federate may receive one or more *Attribute Ownership Divestiture Notification* † invocations for each invocation of this service since different federates may wish to become the owner of different instance attributes.

A request to divest ownership shall remain pending until either the request is granted (via the *Attribute Ownership Divestiture Notification* † service), the requesting federate successfully cancels the request (via the *Cancel Negotiated Attribute Ownership Divestiture* † service), or the federate divests itself of ownership by other means (e.g., the *Attribute Ownership Release Response* or *Unpublish* service). A second negotiated divestiture for an instance attribute already in the process of a negotiated divestiture shall not be legal.

Supplied Arguments

- Object instance designator
- Set of attribute designators
- User-supplied tag

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate owns the specified instance attributes.
- The specified instance attributes are not in the negotiated divestiture process.

Post-conditions

- No change has occurred in instance attribute ownership.
- The RTI has been notified of the federate's request to divest ownership of the specified instance attributes.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate does not own the instance attribute.
- The instance attribute is already in the negotiated divestiture process.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Request Attribute Ownership Assumption* †
- *Attribute Ownership Divestiture Notification* †

— *Attribute Ownership Acquisition Notification †*

7.4 Request Attribute Ownership Assumption †

The *Request Attribute Ownership Assumption* † service shall inform the federate that the specified instance attributes are available for transfer of ownership to the federate. The RTI shall supply an object instance designator and set of attribute designators. The federate may return a subset of the supplied attribute designators for which it is willing to assume ownership via the *Attribute Ownership Acquisition* service or via the *Request Attribute Ownership If Available* service. In the case that the supplied instance attributes are unowned as a result of a federate invoking the *Unconditional Attribute Ownership Divestiture* service, the divesting federate shall not be asked to assume ownership.

Supplied Arguments

- Object instance designator
- Set of attribute designators
- User-supplied tag

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate is publishing the corresponding class attributes at the known class of the specified object instance.
- The federate does not own the specified instance attributes.

Post-conditions

- Instance attribute ownership has not changed.
- The federate has been informed of the set of instance attributes for which the RTI is requesting that the federate assume ownership.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate already owns the instance attribute.
- The federate is not publishing the class attribute at the known class of the object instance.
- Federate internal error

Related Services

- *Unconditional Attribute Ownership Divestiture*
- *Negotiated Attribute Ownership Divestiture*
- *Attribute Ownership Divestiture Notification* †
- *Attribute Ownership Acquisition Notification* †

7.5 Attribute Ownership Divestiture Notification †

The *Attribute Ownership Divestiture Notification* † service shall notify the federate that it no longer owns the specified set of instance attributes. Upon this notification, the federate shall stop updating the specified instance attribute values. The federate may receive multiple notifications for a single invocation of the *Negotiated Attribute Ownership Divestiture* service since different federates may wish to become the owner of different instance attributes.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate owns the specified instance attributes.
- The federate has previously attempted to divest ownership of the specified instance attributes and has not subsequently canceled that request.

Post-conditions

- The federate does not own the specified instance attributes.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate does not own the instance attribute.
- The federate had not previously attempted to divest ownership of the instance attribute.
- Federate internal error

Related Services

- *Negotiated Attribute Ownership Divestiture*
- *Request Attribute Ownership Assumption* †
- *Attribute Ownership Acquisition Notification* †

7.6 Attribute Ownership Acquisition Notification †

The *Attribute Ownership Acquisition Notification* † service shall notify the federate that it now owns the specified set of instance attributes. The federate may then begin updating those instance attribute values. The federate may receive multiple notifications for a single invocation of the *Attribute Ownership Acquisition* service since the federate may wish to become the owner of instance attributes owned by different federates.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate is publishing the corresponding class attributes at the known class of the specified object instance.
- A federate has previously attempted to acquire ownership of the specified instance attributes.
- The specified instance attributes are not owned by any federate in the federation execution.

Post-conditions

- The federate owns the specified instance attributes.
- The federate may stop publishing the corresponding class attributes at the known class of the specified object instance.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate had not previously attempted to acquire ownership of the instance attribute.
- The federate already owns the instance attribute.
- The federate is not publishing the class attribute at the known class of the object instance.
- Federate internal error

Related Services

- *Unconditional Attribute Ownership Divestiture*
- *Negotiated Attribute Ownership Divestiture*
- *Request Attribute Ownership Assumption* †
- *Attribute Ownership Divestiture Notification* †

7.7 Attribute Ownership Acquisition

The *Attribute Ownership Acquisition* service shall request the ownership of the specified instance attributes of the specified object instance. If a specified instance attribute is owned by another federate, the RTI shall invoke the *Request Attribute Ownership Release* † service for that instance attribute at the owning federate. The federate may receive one or more *Attribute Ownership Acquisition Notification* † invocations for each invocation of this service.

A request to acquire ownership shall remain pending until either the request is granted (via the *Attribute Ownership Acquisition Notification* † service) or the requesting federate successfully cancels the request (via the *Cancel Attribute Ownership Acquisition* and *Confirm Attribute Ownership Acquisition Cancellation* † services).

Supplied Arguments

- Object instance designator
- Set of attribute designators
- User-supplied tag

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate is publishing the corresponding class attributes at the known class of the specified object instance.
- The federate does not own the specified instance attributes.

Post-conditions

- The RTI has been informed of the federate's request to acquire ownership of the specified instance attributes.
- The federate shall not stop publishing the corresponding class attributes at the known class of the specified object instance.

Exceptions

- The object instance is not known.
- The federate is not publishing the object class.
- The class attribute is not available at the known class of the object instance.
- The federate is not publishing the class attribute at the known class of the object instance.
- The federate already owns the instance attribute.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Request Attribute Ownership Release* †
- *Attribute Ownership Acquisition Notification* †
- *Cancel Attribute Ownership Acquisition*

— *Cancel Attribute Ownership Acquisition Confirmation*

7.8 Attribute Ownership Acquisition If Available

The *Attribute Ownership Acquisition If Available* service shall request the ownership of the specified instance attributes of the specified object instance only if the instance attribute is unowned by all federates or it is in the process of being divested by its owner. If a specified instance attribute is owned by another federate, the RTI shall not invoke the *Request Attribute Ownership Release* † service for that instance attribute at the owning federate. The federate shall receive either an *Attribute Ownership Acquisition Notification* † or an *Attribute Ownership Unavailable* † invocation for each of the specified instance attributes.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate is publishing the corresponding class attributes at the known class of the specified object instance.
- The federate does not own the specified instance attributes.
- For each of the specified instance attributes, it is not the case that the federate has invoked the *Attribute Ownership Acquisition* service, but has not yet received an invocation of either the *Confirm Attribute Ownership Acquisition Cancellation* † service or the *Attribute Ownership Acquisition Notification* † service.

Post-conditions

- The RTI has been informed of the federate's request to acquire ownership of the specified instance attributes. The federate shall not stop publishing the corresponding class attributes at the known class of the specified object instance.

Exceptions

- The object instance is not known.
- The federate is not publishing the object class.
- The class attribute is not available at the known class of the object instance.
- The federate is not publishing the class attribute at the known class of the object instance.
- The federate already owns the instance attribute.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error
- The federate is already acquiring the instance attribute.

Related Services

- *Attribute Ownership Acquisition Notification* †
- *Attribute Ownership Unavailable* †

7.9 Attribute Ownership Unavailable †

The *Attribute Ownership Unavailable* † service shall inform the federate that the specified instance attributes were not available for ownership acquisition.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate had requested ownership acquisition (if available) for the specified instance attributes.
- The federate does not own the specified instance attributes.

Post-conditions

- The federate has been informed that the specified instance attributes were not available for ownership acquisition.
- The federate may stop publishing the corresponding class attributes at the known class of the specified object instance.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate already owns the instance attribute.
- The federate had not requested ownership acquisition (if available) for the instance attribute.
- Federate internal error

Related Services

- *Attribute Ownership Acquisition If Available*

7.10 Request Attribute Ownership Release †

The *Request Attribute Ownership Release* † service shall request that the federate release ownership of the specified instance attributes of the specified object instance. The *Request Attribute Ownership Release* † service shall provide an object instance designator and set of attribute designators and shall be invoked only as the result of an *Attribute Ownership Acquisition* service invocation by some other federate. The federate may return the subset of the supplied instance attributes for which it is willing to release ownership via the *Attribute Ownership Release Response* service, the *Unconditional Attribute Ownership Divestiture* service, or the *Negotiated Attribute Ownership Divestiture* service.

Supplied Arguments

- Object instance designator
- Set of attribute designators
- User-supplied tag

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate owns the specified instance attributes.

Post-conditions

- The federate has been informed of the set of instance attributes for which the RTI is requesting the federate to release ownership.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate does not own the instance attribute.
- Federate internal error

Related Services

- *Attribute Ownership Acquisition*
- *Attribute Ownership Acquisition Notification* †

7.11 Attribute Ownership Release Response

The *Attribute Ownership Release Response* service shall notify the RTI that the federate is willing to release ownership of the specified instance attributes for the specified object instance. The federate shall use this service to provide an answer to the question posed as a result of the RTI invocation of *Request Attribute Ownership Release* †. The returned argument shall indicate the instance attributes for which ownership was actually released. Completion of the invocation of this service can be viewed as an implied *Attribute Ownership Divestiture Notification* † invocation for all of the instance attributes in the returned argument.

Supplied Arguments

- Object instance designator
- Set of attribute designators for which the federate is willing to release ownership

Returned Arguments

- Set of attribute designators for which ownership is actually released

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate owns the specified instance attributes.
- The federate has been asked to release the specified instance attributes.

Post-conditions

- Ownership is released for the instance attributes in the returned parameter set.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate does not own the instance attribute.
- The federate had not previously been asked to release ownership of the instance attribute.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Request Attribute Ownership Release* †

7.12 Cancel Negotiated Attribute Ownership Divestiture

The *Cancel Negotiated Attribute Ownership Divestiture* service shall notify the RTI that the federate no longer wants to divest ownership of the specified instance attributes.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate owns the specified instance attributes.
- The specified instance attributes were candidates for divestiture.

Post-conditions

- The specified instance attributes are unavailable for divestiture.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate does not own the instance attribute.
- The instance attribute was not a candidate for divestiture.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Negotiated Attribute Ownership Divestiture*

7.13 Cancel Attribute Ownership Acquisition

The *Cancel Attribute Ownership Acquisition* service shall notify the RTI that the federate no longer wants to acquire ownership of the specified instance attributes.

This service shall always receive one of two replies from the RTI. The first form of reply, *Cancel Attribute Ownership Acquisition Confirmation*, shall indicate that the request to acquire ownership of the specified instance attributes has been successfully canceled. The second form of reply, *Attribute Ownership Acquisition Notification* †, shall indicate that the request to acquire ownership of the specified instance attributes was not canceled in time and that the federate has acquired ownership of the instance attributes.

The federate may receive both forms of reply in response to a single *Cancel Attribute Ownership Acquisition* service invocation since the cancellation may succeed for some of the supplied instance attributes and fail for others.

This service shall be used only to cancel requests to acquire ownership of instance attributes that were made via the *Attribute Ownership Acquisition* service. Requests made via the *Attribute Ownership Acquisition If Available* service shall not be explicitly canceled. They may, however, be overridden by an invocation of the *Attribute Ownership Acquisition* service.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The federate does not own the specified instance attributes.
- The federate is attempting to acquire ownership of the specified instance attributes.

Post-conditions

- The RTI has been notified that federate no longer wants to acquire ownership of the specified instance attributes.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate already owns the instance attribute.
- The federate was not attempting to acquire ownership of the instance attribute.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Attribute Ownership Acquisition*
- *Attribute Ownership Acquisition Notification* †

— *Cancel Attribute Ownership Acquisition Confirmation*

7.14 Confirm Attribute Ownership Acquisition Cancellation †

The *Confirm Attribute Ownership Acquisition Cancellation* † service shall inform the federate that the specified instance attributes are no longer candidates for ownership acquisition.

Supplied Arguments

- Object instance designator
- Set of attribute designators

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The federate had attempted to cancel an ownership acquisition request for the specified instance attributes.
- The federate does not own the specified instance attributes.

Post-conditions

- The specified instance attributes are no longer candidates for acquisition by the federate.
- The federate may stop publishing the corresponding class attributes at the known class of the specified object instance.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- The federate already owns the instance attribute.
- The federate had not canceled an ownership acquisition request for the instance attribute.
- Federate internal error

Related Services

- *Cancel Attribute Ownership Acquisition*

7.15 Query Attribute Ownership

The *Query Attribute Ownership* service shall be used to determine the owner of the specified instance attribute. The RTI shall provide the instance attribute owner information via the *Inform Attribute Ownership* † service invocation.

Supplied Arguments

- Object instance designator
- Attribute designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The corresponding class attribute is an available attribute of the known class of the specified object instance.

Post-conditions

- The request for instance attribute ownership information has been received by the RTI.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Inform Attribute Ownership* †

7.16 Inform Attribute Ownership †

The *Inform Attribute Ownership* † service shall be used to provide ownership information for the specified instance attribute. This service shall be invoked by the RTI in response to a *Query Attribute Ownership* service invocation by a federate. This service shall provide the federate a designator of the instance attribute owner (if the instance attribute is owned) or an indication that the instance attribute is available for acquisition.

Supplied Arguments

- Object instance designator
- Attribute designator
- Ownership designator (could be a federate, MOM, or unowned)

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate knows about the object instance with the specified designator.
- The corresponding class attribute is an available attribute of the known class of the specified object instance.
- The federate has previously invoked the *Query Attribute Ownership* service and has not yet received an *Inform Attribute Ownership* † service invocation in response.

Post-conditions

- The federate has been informed of the instance attribute ownership.

Exceptions

- The object instance is not known.
- The attribute designator is not recognized.
- Federate internal error

Related Services

- *Query Attribute Ownership*

7.17 Is Attribute Owned By Federate

The *Is Attribute Owned By Federate* service shall be used to determine if the specified instance attribute of the specified object instance designator is owned by the invoking federate. The service shall return a Boolean value indicating ownership status of the specified instance attribute.

Supplied Arguments

- Object instance designator
- Attribute designator

Returned Arguments

- Instance attribute ownership indicator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The corresponding class attribute is an available attribute of the known class of the specified object instance.

Post-conditions

- The federate has the requested ownership information.

Exceptions

- The object instance is not known.
- The class attribute is not available at the known class of the object instance.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- None

8. Time management

8.1 Overview

Time in the system being modeled shall be represented in the federation as points along a federation time axis. Each federate may advance along the axis during the course of the execution. Such federate time advances may be constrained by the progress of other federates or unconstrained.

Time management is concerned with the mechanisms for controlling the advancement of each federate along the federation time axis. In general, time advances shall be coordinated with object management services so that information is delivered to federates in a causally correct and ordered fashion.

A federate that becomes time regulating may associate some of its activities (such as updating instance attribute values and sending interactions) with points on the federation time axis. It shall do so by assigning time stamps to activities that reflect the points on the federation time axis with which the activities are associated. A federate that is time constrained is interested in receiving notifications of these activities (such as reflecting instance attribute values and receiving interactions) in a federation-wide time-stamp order. Use of the time management services allows this type of coordination among time-regulating and time-constrained federates in an execution. The coordination shall be achieved by various constraints on federate activities described in this chapter.

The activities of federates that are neither time regulating nor time constrained (the default state of all federates upon joining an execution) shall not be coordinated with other federates by the RTI, and such federates need not make use of any of the time management services.

8.1.1 Messages

The manner in which HLA services are coordinated with time shall be through the concept of *messages*.

- Invocation of the *Update Attribute Values* service, *Send Interaction* service, *Send Interaction with Region* service, or *Delete Object Instance* service by a federate shall be called *sending a message*.
- Invocation of the *Reflect Attribute Values †* service, *Receive Interaction †* service, or *Remove Object Instance †* service at a federate shall be called *receiving a message*.

Messages sent by one federate typically result in one or more other federates receiving a corresponding message. The mapping from one sent message to one or more received messages shall follow the rules of Clauses 5, 6, and 9. For example, a sent message representing an *Update Attribute Values* service invocation shall result only in received messages representing *Reflect Attribute Values †* service invocations at the appropriate federates depending on the normal publication/subscription rules. Messages shall also be referred to as *events*.

Each message, sent or received, shall be either a time-stamped order (TSO) message or a receive order (RO) message. The order type of a message shall be determined by several factors:

- *Preferred order type*: The preferred order type of a message shall be the same as the preferred order type of the data contained in the message (instance attribute values or interactions). Each class attribute and interaction class shall be provided with a preferred order type in the FED that indicates the order type (TSO or RO) that should be used when sending messages carrying values for instances of these classes. In the case of sent messages representing a *Delete Object Instance* service invocation, the preferred order type of the message shall be based on the preferred order

- type of the `privilegeToDeleteObject` attribute of the specified object instance. Federates may use the *Change Attribute Order Type* service to change the preferred order type of instance attributes; the preferred order type of class attributes may not be changed during an execution. Federates may use the *Change Interaction Order Type* service to change the preferred order type of interaction classes.
- *Presence of a time stamp:* Each of the services that corresponds to sending or receiving a message shall have an optional time-stamp argument. If a message is sent using a service invocation in which the optional time stamp is supplied, then the federate is attempting to send a TSO message. If a message is sent and the optional time stamp is not supplied, then the federate is attempting to send an RO message. All received TSO messages shall have time stamps; all received RO messages shall not have time stamps.
 - *Federate's time status:* Whether or not a federate is time regulating shall determine whether or not a federate can send TSO messages. Similarly, whether or not a federate is time constrained shall determine whether or not the federate can receive TSO messages.
 - *Sent message order type:* The order type of a received message shall depend on the order type of the corresponding sent message.

These factors shall be considered together when determining if a given message is sent or received as a TSO message or as an RO message.

The order type of a sent message shall be determined by the preferred order type of the message at the sending federate, whether or not that federate is time regulating, and whether or not a time stamp was used in the service invocation that sends the message. The following table shall illustrate how the order type of a sent message shall be determined.

Table 1—Order type of a sent message

Preferred order type?	Sending federate is time regulating?	Time stamp was used?	Order type of sent message
RO	No	No	RO
RO	No	Yes	RO ^a
RO	Yes	No	RO
RO	Yes	Yes	RO ^a
TSO	No	No	RO
TSO	No	Yes	RO ^a
TSO	Yes	No	RO
TSO	Yes	Yes	TSO

^a Despite the presence of a time stamp, messages shall be RO if the preferred order type is RO or the sending federate is not time regulating. If a time stamp is provided by the sending federate, it will be removed.

The order type of a received message shall be determined by whether or not that federate is time constrained and by the order type of the corresponding sent message. The following table shall illustrate how the order type of a received message shall be determined:

Table 2—Order type of a received message

Receiving federate is time constrained?	Order type of corresponding sent message?	Order type of received message?
No	RO	RO
No	TSO	RO
Yes	RO	RO
Yes	TSO	TSO

Because of the above rule defining the order type of a received message, the RTI will sometimes convert a sent TSO message to a received RO message at some receiving federates. The need for such conversions shall be considered on a per-federate basis, and the received messages at different federates that correspond to the same sent message may be of different order types. Sent RO messages shall never be converted to received TSO messages.

Messages that are received as TSO messages shall be received only by a given federate in time-stamp order, regardless of the federates from which the messages originate and regardless of the order in which the messages were sent. Thus two TSO messages with different time stamps shall always be received by each federate in the same order. Multiple TSO messages having the same time stamp shall be received in an indeterminate order.

Messages that are received as RO messages shall be received in an arbitrary order.

8.1.2 Logical time

Each federate, upon joining an execution, shall be assigned a *logical time*. A federate's logical time shall initially be set to the initial time on the federation time axis (time zero). Time within a federation shall only advance; thus a federate may request to advance only to a time that is greater than or equal to its current logical time. In order for a federate to advance its logical time, it shall request an advance explicitly. The advance shall not occur until the RTI issues a grant. In general, at any instant during an execution, different federates may be at different logical times.

Federates also may become time regulating and/or time constrained. The logical times of federates that are time regulating shall be used to constrain the advancement of the logical times of federates that are time constrained.

8.1.3 Time-regulating federates

Only time regulating federates may send TSO messages. A federate shall request to become time regulating by invoking the *Enable Time Regulation* service. The RTI shall subsequently make the federate time regulating by invoking the *Time Regulation Enabled* † service at that federate. A federate shall cease to be time regulating whenever it invokes the *Disable Time Regulation* service.

Each time-regulating federate shall provide a *lookahead* value when becoming time regulating. Lookahead shall be a non-negative value that establishes a lower bound on the time stamps that can be sent in TSO messages by the federate. Specifically, a time-regulating federate shall not send a TSO message that contains a time stamp less than its current logical time plus its lookahead. Once established, a federate's lookahead value may be changed only using the *Modify Lookahead* service.

A time-regulating federate with a lookahead value of zero shall be subject to an additional restriction. If such a federate has advanced its logical time by use of *Time Advance Request* or *Next Event Request*, then it shall not send TSO messages that contain time stamps less than *or equal to* its logical time (rather than the usual less-than restriction). Subsequent use of a different time advancement service shall lift this additional restriction.

NOTE—A time-regulating federate need not send TSO messages in time-stamp order, but all TSO messages that it sends shall be received by other federates in time-stamp order (if they are received as TSO messages).

8.1.4 Time-constrained federates

Only time-constrained federates can receive TSO messages. A federate shall request to become time constrained by invoking the *Enable Time Constrained* service. The RTI shall subsequently make the federate time constrained by invoking the *Time Constrained Enabled* † service at that federate. A federate shall cease to be time constrained whenever it invokes the *Disable Time Constrained* service.

Each federate in an execution, whether time constrained or not, shall have an associated lower bound on the time stamp (*LBTS*) value. The LBTS value shall be calculated by the RTI and shall represent the smallest time stamp that could ever be received by that federate in a TSO message if that federate were time constrained. In performing this calculation for a given federate, the RTI shall take into account the logical time and lookahead of all time-regulating federates in the execution (less the given federate if it is also time regulating) to determine the smallest time stamp that the given federate could receive in a TSO message. If there are no time-regulating federates in an execution (less the given federate), then that federate's LBTS value shall be infinite.

To help ensure that time-constrained federates receive all TSO messages in time-stamp order, a time-constrained federate shall not be permitted to advance its logical time beyond its LBTS value. This ensures that a time-constrained federate cannot receive a TSO message with a time stamp that is less than the federate's logical time. Should a time-constrained federate request to advance its logical time beyond its current LBTS value, the time advance shall not be granted until the federate's LBTS has increased sufficiently for the constraint to be met.

8.1.5 Advancing time

A federate may advance its logical time only by requesting a time advancement from the RTI. Its logical time shall not actually be advanced until the RTI responds with a *Time Advance Grant* † service invocation at that federate. The interval between these service invocations shall be the Time Advancing state; this is shown in the statechart in Figure 12.

A federate shall request to advance its logical time by invoking one of the following services:

- *Time Advance Request*
- *Time Advance Request Available*
- *Next Event Request*
- *Next Event Request Available*
- *Flush Queue Request*

Each service shall take a requested logical time as an argument, shall request slightly different coordination from the RTI, and shall be further elaborated in the service descriptions as described in the following table.

Table 3—Service descriptions

	Constraint on advance to t_1	Messages delivered before grant to t_2	Constraint on grant to t_2	
TAR	Can't send $ts < t_1 + \text{lookahead}$	All queued RO messages All TSO messages with $ts \leq t_2$	Can't send $ts < t_2 + \text{lookahead}$	$t_2 = t_1$
TAR (zero lookahead)	Can't send $ts \leq t_1$	All queued RO messages All TSO messages with $ts \leq t_2$	Can't send $ts \leq t_2$	$t_2 = t_1$
TARA	Can't send $ts < t_1 + \text{lookahead}$	All queued RO messages All TSO messages with $ts < t_2$ All queued TSO messages with $ts = t_2$	Can't send $ts < t_2 + \text{lookahead}$	$t_2 = t_1$
NER	Can't send $ts < t_1 + \text{lookahead}$	All queued RO messages Smallest TSO message that will ever be received that has a $ts \leq t_1$ and all other TSO messages with the same ts	Can't send $ts < t_2 + \text{lookahead}$	$t_2 \leq t_1$
NER (zero lookahead)	Can't send $ts \leq t_1$	All queued RO messages Smallest TSO message that will ever be received that has a $ts \leq t_1$ and all other TSO messages with the same ts	Can't send $ts \leq t_2$	$t_2 \leq t_1$
NERA	Can't send $ts < t_1 + \text{lookahead}$	All queued RO messages Smallest TSO message that will ever be received that has a $ts \leq t_1$ and all other queued TSO messages with the same ts	Can't send $ts < t_2 + \text{lookahead}$	$t_2 \leq t_1$
FQR	Can't send $ts < t_1 + \text{lookahead}$	All queued RO messages All queued TSO messages	Can't send $ts < t_2 + \text{lookahead}$	$t_2 \leq t_1$

The *Time Advance Grant* \dagger service shall be used to grant an advance regardless of which form of request was made to advance time. This service shall take a logical time as an argument, and this shall be the federate's new logical time. The guarantee that the RTI makes about message delivery relative to the provided logical time shall depend on the type of request to advance time; the specific guarantees shall be provided in the service descriptions. Note that in some cases, the RTI can advance a federate to a logical time that is less than the time that the federate requested.

The RTI shall grant an advance to logical time T only when it can guarantee that all TSO messages with time stamps less than T (or in some cases less than or equal to T) have been delivered to the federate. This guarantee enables the federate to simulate the behavior of the entities it represents up to logical time T without concern for receiving new events with time stamps less than T . Note that in some cases, providing this guarantee will require the RTI to wait for a significant period of wall-clock time to elapse before it can grant a time advancement to a time-constrained federate. However, in the case of federates that are not time constrained (and thus cannot receive TSO messages), the guarantee is trivially true and the advance can be granted almost immediately.

The advancing of logical time by time-regulating federates is important because it acts as their promise not to send any TSO messages with time stamps less than some specified time. In general, when time-regulating federates move their logical times forward, time-constrained federates can move forward as well.

Federates that are not time regulating need not advance their logical time, but may do so. Such advancements shall have no effect on other federates' time advancement unless the advancing federate later becomes time regulating (at which point the advancing federate will begin to have an effect on the advancement of time-constrained federates).

8.1.6 Putting it all together

The statechart shown in Figure 12 shall illustrate when a federate may become time regulating and time constrained, when time advances may be requested, how a federate enables or disables asynchronous message delivery, and the effect these activities have on determining sent and received message order types and when messages may be sent and received.

The transition labeled "Send Message" shall represent any service invocation that is called sending a message. As represented in the statechart, such a transition can occur at any time and shall result in the federate returning to whatever state it was in before the transition. The column to the right of the statechart elaborates on how the order type of the sent message is determined. Each part of the definition of "Send Message" shall be composed of a conversion rule (denoted as two terms separated by an arrow) and an optional Boolean guard (denoted in square braces, just as in statecharts). The term to the left of the arrow in each conversion rule shall represent the preferred order type of the message and whether or not a time stamp was provided by the invoking federate. The term to the right of the arrow shall represent the order type of the sent message. The guard shall represent under what circumstances the conversion rule applies. So each part of the definition shall be read as: if the preferred order type of the message is as indicated to the left of the arrow, the usage of a time stamp is as described to the left of the arrow, and the Boolean guard (if present) is true, then the order type of the sent message is as indicated to the right of the arrow. The conversion rules provided in the statechart are the same as the results contained in the tables in 8.1.1.

The transitions labeled "Receive Message #1" and "Receive Message #2" shall be read similarly with one exception: The conversion rules shall be slightly different. The term to the left of the arrow shall represent

the order type of the received message. The term to the right of the arrow shall represent the order type of the corresponding sent message.

Federates may send messages at any time in this diagram. If the federate is time regulating and sending a TSO message, the time stamp of that message shall be constrained as described in 8.1.3 with one exception: When a federate is in the Time Advancing state, the stated constraint is not strong enough. Rather than comparing the time stamp of the TSO message to the federate's logical time (plus lookahead), the time stamp shall be compared to the federate's requested logical time (plus its lookahead).¹

When federates are eligible to receive messages shall be dependent on several factors. If the federate is not time constrained, it may receive messages at any time (although only RO messages may be received). If the federate is time constrained, it shall normally receives messages only when in the Time Advancing state. However, federates may enable asynchronous message delivery (via the *Enable Asynchronous Delivery* service), which permits them to receive RO messages (but not TSO messages) when not in the Time Advancing state.

Which RO messages will be received when a federate is eligible to receive RO messages shall depend only on which messages have been sent that will be received as RO messages by that federate. In general, if a federate is eligible to receive RO messages, it may receive all RO messages that it has not yet received.

Which TSO messages will be received when a federate is eligible to receive TSO messages shall depend on which TSO messages have been sent that will be received as TSO messages, what time stamps the messages have, and what form of time advancement was requested. Precisely which TSO messages will be received shall be defined in each of the different time advancement services.

Because messages are not always eligible for delivery, the RTI shall internally queue pending messages for each federate. The RTI shall queue all messages that the federate will receive as TSO or RO messages. When messages are finally delivered to the federate, they shall be removed from the queue.

¹ Note that if the federate is granted to a time that is less than its requested logical time (e.g., the request used the *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service), the constraints shall ease upon leaving the Time Advancing state.

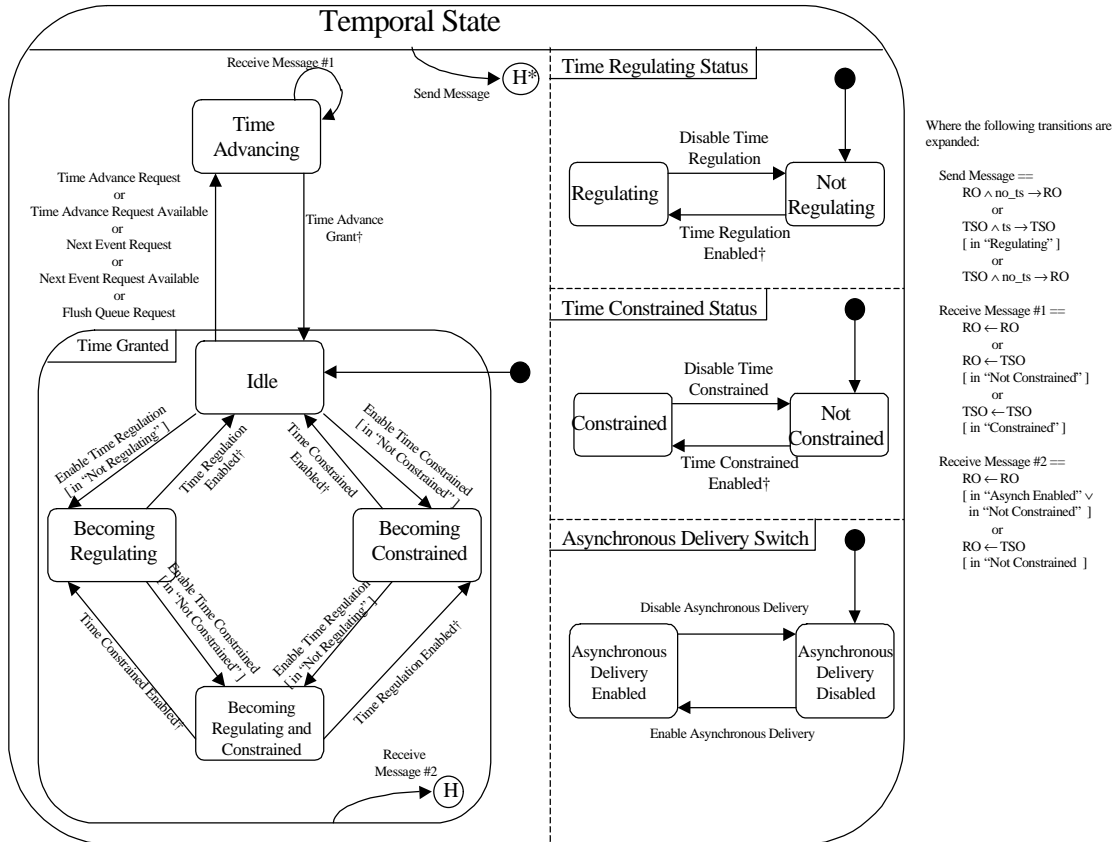


Figure 12—Temporal state

NOTE—failure to make full use of the time management services (and hence causal ordering) can lead to unusual results. For example, if a federate receiving messages concerning a particular object instance is not time constrained, it could receive a message concerning the deletion of that object instance and subsequently receive a message concerning the updating of the value of one of that object instance's attributes. This is because a federate that is not time constrained can receive only RO messages, and RO messages originating from different federates (e.g., one that updates an attribute instance and one that deletes the object instance) are not causally ordered.

8.2 Enable Time Regulation

The *Enable Time Regulation* service shall enable time regulation for the federate invoking the service, thereby enabling the federate to send TSO messages. The federate shall request that its logical time and lookahead value be set to the values specified as arguments. The RTI may not be able to set the federate's logical time to the value that was requested because doing so might enable the federate to, for example, send a message with a time stamp smaller than the current logical time of another federate. The RTI shall indicate the logical time assigned to the federate through the *Time Regulation Enabled* \dagger service. The logical time that is assigned shall be greater than or equal to that requested by the federate.

Because the invocation of this service may require the RTI to advance the invoking federate's logical time, this service has an additional meaning for time-constrained federates. Since the advancing logical time for a time-constrained federate is synonymous with a guarantee that all TSO messages with time stamps less than the new logical time have been delivered, the invocation of this service shall be considered an implicit *Time Advance Request* service invocation. The subsequent invocation of *Time Regulation Enabled* \dagger shall be considered an implicit *Time Advance Grant* \dagger service invocation. Thus if a time-constrained federate attempts to become time regulating, it may receive RO and TSO messages between its invocation of *Enable Time Regulation* and the RTI's invocation of *Time Regulation Enabled* \dagger at the federate. This special case is not illustrated in the statechart in Figure 12.

Supplied Arguments

- Value of federation time
- Lookahead value

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, or *Enable Time Regulation* services is pending.
- Time regulation is not enabled in the federate.
- The specified federation time is greater than or equal to the federate's current logical time.
- If the federate is time constrained, the argument is equal to the federate's current logical time.

Post-conditions

- The RTI is informed of the federate's request to enable time regulation.

Exceptions

- The federate is not a federation execution member.
- Time regulation is already enabled.
- Invalid federation time
- Invalid lookahead time
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.
- An *Enable Time Regulation* request is already pending.
- Save in progress
- Restore in progress

— RTI internal error

Related Services

- *Time Regulation Enabled †*
- *Disable Time Regulation*
- *Enable Time Constrained*
- *Time Constrained Enabled †*
- *Disable Time Constrained*

8.3 Time Regulation Enabled †

Invocation of the *Time Regulation Enabled* † service shall indicate that a prior request to enable time regulation has been honored. The value of this service's argument shall indicate that the logical time of the federate has been set to the specified value.

Supplied Arguments

- Current logical time of the federate

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The *Enable Time Regulation* service is pending.

Post-conditions

- Time regulation is enabled and the federate may now send TSO messages. The federate's logical time shall be set to the value specified as the argument to this service. The federate's lookahead shall be set to that specified in the corresponding *Enable Time Regulation* request.
- If the federate is time constrained, no additional TSO messages shall be delivered with time stamps less than or equal to the provided time.

Exceptions

- Invalid federation time
- *Enable Time Regulation* was not pending.
- Federate internal error

Related Services

- *Enable Time Regulation*
- *Disable Time Regulation*
- *Enable Time Constrained*
- *Time Constrained Enabled* †
- *Disable Time Constrained*

8.4 Disable Time Regulation

Invocation of the *Disable Time Regulation* service shall indicate that the federate is disabling time regulation. Subsequent messages sent by the federate shall be sent automatically as RO messages.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- Time regulation is enabled in the federate.

Post-conditions

- The federate may no longer send TSO messages.

Exceptions

- The federate is not a federation execution member.
- *Time Regulation* was not enabled.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Regulation*
- *Time Regulation Enabled* †
- *Enable Time Constrained*
- *Time Constrained Enabled* †
- *Disable Time Constrained*

8.5 Enable Time Constrained

The *Enable Time Constrained* service shall request that the federate invoking the service become time constrained. The RTI shall indicate that the federate is time constrained by invoking the *Time Constrained Enabled* † service.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, or *Enable Time Constrained* services is pending.
- The federate is not already time constrained.

Post-conditions

- The RTI is informed of the federate's request to become time constrained

Exceptions

- The federate is not a federation execution member.
- Time constrained is already enabled.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.
- An *Enable Time Constrained* request is already pending.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Regulation*
- *Time Regulation Enabled* †
- *Disable Time Regulation*
- *Time Constrained Enabled* †
- *Disable Time Constrained*
- *Enable Asynchronous Delivery*
- *Disable Asynchronous Delivery*

8.6 Time Constrained Enabled †

Invocation of the *Time Constrained Enabled* † service shall indicate that a prior request to become time constrained has been honored. The value of this service's argument shall indicate the current logical time of the federate.

When a federate changes to be time constrained, TSO messages stored in the RTI's internal queues that have time stamps greater than or equal to the federate's logical time shall be delivered in time-stamp order. TSO messages delivered to the federate before it becomes time constrained, possibly including messages with time stamps greater than or equal to the federate's current logical time, shall be delivered as RO messages.

Federates that are time constrained may receive messages only when in the Time Advancing state unless asynchronous message delivery is enabled (by use of the *Enable Asynchronous Delivery* † service). If asynchronous message delivery is enabled, the time-constrained federate may receive RO messages when not in the Time Advancing state, but TSO messages may still be received only when in the Time Advancing state.

If the federate is time regulating, the argument shall equal the federate's current logical time. If the federate is not time regulating, the argument shall be greater than or equal to the federate's current logical time.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The *Enable Time Constrained* service is pending.

Post-conditions

- The federate may now receive TSO messages, and its logical time advances are constrained so that the federate's logical time shall never exceed the LBTS value computed by the RTI for the federate. The federate's logical time shall be set to the value specified as the argument to this service.

Exceptions

- The federation time is invalid.
- *Enable Time Constrained* was not pending.
- Federate internal error

Related Services

- *Enable Time Regulation*
- *Time Regulation Enabled* †
- *Disable Time Regulation*
- *Enable Time Constrained*
- *Disable Time Constrained*
- *Enable Asynchronous Delivery*
- *Disable Asynchronous Delivery*

8.7 Disable Time Constrained

Invocation of the *Disable Time Constrained* service shall indicate that the federate is no longer time constrained. All enqueued and subsequent TSO messages shall be delivered to the federate as RO messages.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate is time constrained.

Post-conditions

- The federate is no longer time constrained and can no longer receive TSO messages.

Exceptions

- The federate is not a federation execution member.
- *Time Constrained* was not enabled.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Regulation*
- *Time Regulation Enabled* †
- *Disable Time Regulation*
- *Enable Time Constrained*
- *Time Constrained Enabled* †
- *Enable Asynchronous Delivery*
- *Disable Asynchronous Delivery*

8.8 Time Advance Request

The *Time Advance Request* service shall request an advance of the federate's logical time and release zero or more messages for delivery to the federate.

Invocation of this service shall cause the following set of messages to be delivered to the federate:

- All messages queued in the RTI that the federate will receive as RO messages.
- All messages that the federate will receive as TSO messages that have time stamps less than or equal to the specified time.

After invoking *Time Advance Request*, the messages shall be passed to the federate by the RTI invoking the *Receive Interaction* †, *Reflect Attribute Values* †, and *Remove Object Instance* † services.

By invoking *Time Advance Request* with the specified time, the federate is guaranteeing that it will not generate a TSO message at any time in the future with a time stamp less than or equal to the specified time, even if the federate's lookahead is zero. Further, the federate may not generate any TSO messages in the future with time stamps less than the specified time plus that federate's current lookahead.

A *Time Advance Grant* † shall complete this request and indicate to the federate that it has advanced its logical time to the specified time, and that no additional TSO messages will be delivered to the federate in the future with time stamps less than or equal to the time of the grant.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified time is greater than or equal to the federate's logical time.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, *Enable Time Regulation*, or *Enable Time Constrained* services is pending.

Post-conditions

- The federate may not send any TSO messages with time stamps less than the specified time plus the federate's actual lookahead.
- If the federate's lookahead is zero, it may not send any TSO messages with time stamps less than or equal to the specified time.
- The RTI is informed of the federate's request to advance time.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.
- *Enable Time Regulation* request is already pending.
- *Enable Time Constrained* request is already pending.
- The federate is not a federation execution member.
- Save in progress

- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request Available*
- *Next Event Request*
- *Next Event Request Available*
- *Flush Queue Request*
- *Time Advance Grant* †

8.9 Time Advance Request Available

The *Time Advance Request Available* service shall request an advance of the federate's logical time. It is similar to *Time Advance Request* to time *T* except

- The RTI shall not guarantee delivery of all messages with time stamps equal to *T* when a Time Advance Grant \dagger to time *T* is issued, and
- After the federate receives a Time Advance Grant \dagger to time *T*, it can send additional messages with time stamps equal to *T* if the federate's lookahead value is zero.

Invocation of this service shall cause the following set of messages to be delivered to the federate:

- All messages queued in the RTI that the federate will receive as RO messages.
- All messages that the federate will receive as TSO messages that have time stamps less than the specified time.
- Any messages queued in the RTI that the federate will receive as TSO messages that have time stamps equal to the specified time.

After invoking *Time Advance Request Available*, the messages shall be passed to the federate by the RTI invoking the *Receive Interaction* \dagger , *Reflect Attribute Values* \dagger , and *Remove Object Instance* \dagger services.

By invoking *Time Advance Request Available* with the specified time, the federate is guaranteeing that it will not generate a TSO message at any time in the future with a time stamp less than the specified time, plus that federate's current lookahead.

A *Time Advance Grant* \dagger shall complete this request and indicate to the federate that it has advanced its logical time to the specified time, and no additional TSO messages shall be delivered to the federate in the future with time stamps less than the time of the grant. Additional messages with time stamps equal to the time of the grant can arrive in the future.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified time is greater than or equal to the federate's logical time.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, *Enable Time Regulation*, or *Enable Time Constrained* services is pending.

Post-conditions

- The federate may not send any TSO messages with time stamps less than the specified time plus the federate's actual lookahead.
- The RTI is informed of the federate's request to advance time.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.

- *Enable Time Regulation* request is already pending.
- *Enable Time Constrained* request is already pending.
- Federation time has already passed.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Next Event Request*
- *Next Event Request Available*
- *Flush Queue Request*
- *Time Advance Grant* †

8.10 Next Event Request

The *Next Event Request* service shall request the logical time of the federate to be advanced to the time stamp of the next TSO message that will be delivered to the federate, provided that message has a time stamp no greater than the logical time specified in the request.

Invocation of this service shall cause the following set of messages to be delivered to the federate:

- All messages queued in the RTI that the federate will receive as RO messages
- The smallest time-stamped message that will ever be received by the federate as a TSO message with a time stamp less than or equal to the specified time, and all other messages containing the same time stamp that the federate will receive as TSO messages

After invocation of *Next Event Request*, the messages shall be passed to the federate by the RTI invoking the *Receive Interaction* †, *Reflect Attribute Values* †, and *Remove Object Instance* † services.

By invoking *Next Event Request* with the specified time, the federate is guaranteeing that it will not generate a TSO message before the pending *Time Advance Grant* † invocation with a time stamp less than or equal to the specified time (or less than the specified time plus the federate's lookahead if its lookahead is not zero).

If it does not receive any TSO messages before the *Time Advance Grant* † invocation, the federate shall guarantee that it will not generate a TSO message at any time in the future with a time stamp less than or equal to the specified time (or less than the specified time plus the federate's lookahead if its lookahead is not zero).

If it does receive any TSO messages before the *Time Advance Grant* † invocation, the federate shall guarantee that it will not generate a TSO message at any time in the future with a time stamp less than or equal to the time of the grant (or less than the time of the grant plus the federate's lookahead if its lookahead is not zero).

A *Time Advance Grant* † shall complete this request and indicate to the federate that it has advanced its logical time to the time stamp of the TSO messages that are delivered, if any, or to the specified time if no TSO messages were delivered. It shall also indicate that no TSO messages will be delivered to the federate in the future with time stamps less than or equal to the time of the grant.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified time is greater than or equal to the federate's logical time.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, *Enable Time Regulation*, or *Enable Time Constrained* services is pending.

Post-conditions

- The federate may not send any TSO messages with time stamps less than the specified time plus the

- federate's actual lookahead.
- If the federate's lookahead is zero, it may not send any TSO messages with time stamps less than or equal to the specified time.
- The RTI is informed of the federate's request to advance time.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.
- *Enable Time Regulation* request is already pending.
- *Enable Time Constrained* request is already pending.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Time Advance Request Available*
- *Next Event Request Available*
- *Flush Queue Request*
- *Time Advance Grant* †

8.11 Next Event Request Available

The *Next Event Request Available* service shall request the logical time of the federate to be advanced to the time stamp of the next TSO message that will be delivered to the federate, provided that message has a time stamp no greater than the logical time specified in the request. It is similar to *Next Event Request* except

- The RTI shall not guarantee delivery of all messages with time stamps equal to T when a *Time Advance Grant* \dagger to time T is issued, and
- After the federate receives a *Time Advance Grant* \dagger to time T , it can send additional messages with time stamps equal to T if the federate's lookahead value is zero.

Invocation of this service shall cause the following set of messages to be delivered to the federate:

- All messages queued in the RTI that the federate will receive as RO messages
- The smallest time-stamped message that will ever be received by the federate as a TSO message with a time stamp less than or equal to the specified time, and any other messages queued in the RTI that the federate will receive as TSO messages and that have the same time stamp.

After invoking *Next Event Request Available*, the messages shall be passed to the federate by the RTI invoking the *Receive Interaction* \dagger , *Reflect Attribute Values* \dagger , and *Remove Object Instance* \dagger services.

By invoking *Next Event Request Available* with the specified time, the federate is guaranteeing that it will not generate a TSO message before the pending *Time Advance Grant* \dagger invocation with a time stamp less than the specified time plus the federate's lookahead.

If it does not receive any TSO messages before the *Time Advance Grant* \dagger invocation, the federate shall guarantee that it will not generate a TSO message at any time in the future with a time stamp less than the specified time plus the federate's lookahead.

If it does receive any TSO messages before the *Time Advance Grant* \dagger invocation, the federate shall guarantee that it will not generate a TSO message at any time in the future with a time stamp less than the time of the grant plus the federate's lookahead.

A *Time Advance Grant* \dagger shall complete this request and indicate to the federate that it has advanced its logical time to the time stamp of the TSO messages that are delivered, if any, or to the specified time if no TSO messages were delivered. A *Time Advance Grant* \dagger shall also indicate that no TSO messages will be delivered to the federate in the future with time stamps less than the time of the grant.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified time is greater than or equal to the federate's logical time.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, *Enable Time Regulation*, or *Enable Time Constrained* services is pending.

Post-conditions

- The federate may not send TSO messages with time stamps less than the specified time plus the federate's actual lookahead.
- The RTI is informed of the federate's request to advance time.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending
- *Enable Time Regulation* request is already pending.
- *Enable Time Constrained* request is already pending.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Time Advance Request Available*
- *Next Event Request*
- *Flush Queue Request*
- *Time Advance Grant* †

8.12 Flush Queue Request

The *Flush Queue Request* service shall request that all messages queued in the RTI that the federate will receive as TSO messages be delivered now. The RTI shall deliver all such messages as soon as possible, despite the fact that it may not be able to guarantee that no future messages containing smaller time stamps could arrive. If the federate will not receive any additional TSO messages with time stamps less than the specified time, the federate's logical time shall be advanced to the specified time. Otherwise, the RTI shall advance the federate's logical time as far as possible, but potentially not at all.

Invocation of this service shall cause the following set of messages to be delivered to the federate:

- All messages queued in the RTI that the federate will receive as RO messages.
- All messages queued in the RTI that the federate will receive as TSO messages.

After invoking *Flush Queue Request*, the messages shall be passed to the federate by the RTI invoking the *Receive Interaction* †, *Reflect Attribute Values* †, and *Remove Object Instance* † services.

By invoking *Flush Queue Request* with the specified time, the federate is guaranteeing that it will not generate a TSO message before the pending *Time Advance Grant* † invocation with a time stamp less than the specified time plus the federate's lookahead.

After the *Time Advance Grant* † invocation, the federate shall guarantee that it shall not generate a TSO message at any time in the future with a time stamp less than the time of the grant plus the federate's lookahead.

A *Time Advance Grant* † shall complete this request and indicate to the federate that it has advanced its logical time to the time of the grant, and no additional TSO messages shall be delivered to the federate in the future with time stamps less than the time of the grant.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The specified time is greater than or equal to the federate's logical time.
- None of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, *Flush Queue Request*, *Enable Time Regulation*, or *Enable Time Constrained* services is pending.

Post-conditions

- The federate may not send any TSO messages with time stamps less than the specified time plus the federate's actual lookahead.
- The RTI is informed of the federate's request to advance time.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service is already pending.

- *Enable Time Regulation* request is already pending.
- *Enable Time Constrained* request is already pending.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Time Advance Request Available*
- *Next Event Request*
- *Next Event Request Available*
- *Time Advance Grant* †

8.13 Time Advance Grant †

Invocation of the *Time Advance Grant* † service shall indicate that a prior request to advance the federate's logical time has been honored. The argument of this service shall indicate that the logical time for the federate has been advanced to this value.

If the grant is issued in response to invocation of *Next Event Request* or *Time Advance Request*, the RTI shall guarantee that no additional TSO messages shall be delivered in the future with time stamps less than or equal to this value.

If the grant is in response to an invocation of *Time Advance Request Available*, *Next Event Request Available*, or *Flush Queue Request*, the RTI shall guarantee that no additional TSO messages shall be delivered in the future with time stamps less than the value of the grant.

Supplied Arguments

- Value of federation time

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- One of the *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* services is pending.

Post-conditions

- If the federate has a change to its lookahead value pending, its new actual lookahead value shall be equal to the maximum of the federate's requested lookahead and the federate's actual lookahead less the amount of time advanced (the federate's old logical time less the provided logical time).
- If *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* has been invoked, the federate may not send TSO messages with time stamps less than the provided time plus the federate's actual lookahead.
- If *Next Event Request* has been invoked and the federate's actual lookahead is zero, the federate may not send TSO messages with time stamps less than or equal to the provided time.
- No additional TSO messages shall be delivered with time stamps less than or equal to the provided time if *Time Advance Request* or *Next Event Request* has been invoked, or with time stamps less than the provided time if *Time Advance Request Available*, *Next Event Request Available*, or *Flush Queue Request* has been invoked.

Exceptions

- The federation time is invalid.
- The *Time Advance Request*, *Time Advance Request Available*, *Next Event Request*, *Next Event Request Available*, or *Flush Queue Request* service was not pending.
- Federate internal error

Related Services

- *Time Advance Request*
- *Time Advance Request Available*
- *Next Event Request*

- *Next Event Request Available*
- *Flush Queue Request*

8.14 Enable Asynchronous Delivery

Invocations of the *Enable Asynchronous Delivery* service shall instruct the RTI to deliver received RO messages to the invoking federate when it is in either the Time Advancing or Time Granted state.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- Asynchronous delivery is disabled at the federate.

Post-conditions

- Asynchronous delivery is enabled at the federate.

Exceptions

- The federate is not a federation execution member.
- Asynchronous delivery is already enabled.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Constrained*
- *Time Constrained Enabled* †
- *Disable Time Constrained*
- *Disable Asynchronous Delivery*

8.15 Disable Asynchronous Delivery

Invocations of the *Disable Asynchronous Delivery* service shall instruct the RTI to deliver received RO messages to the invoking federate only when it is in the Time Advancing state and the federate is time constrained.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- Asynchronous delivery is enabled at the federate.

Post-conditions

- Asynchronous delivery is disabled at the federate.

Exceptions

- The federate is not a federation execution member.
- Asynchronous delivery is already disabled.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Time Constrained*
- *Time Constrained Enabled* †
- *Disable Time Constrained*
- *Enable Asynchronous Delivery*

8.16 Query LBTS

The *Query LBTS* service shall request the invoking federate's current value of LBTS.

Supplied Arguments

- None

Returned Arguments

- Current value of invoking federate's LBTS

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate receives the current value of its LBTS.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Query Federate Time*
- *Query Minimum Next Event Time*

8.17 Query Federate Time

The *Query Federate Time* service shall request the current value of the invoking federate's logical time.

Supplied Arguments

- None

Returned Arguments

- Current value of invoking federate's logical time

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate receives the current value of its logical time.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Query LBTS*
- *Query Minimum Next Event Time*

8.18 Query Minimum Next Event Time

The *Query Minimum Next Event Time* service shall request the minimum of LBTS and the time stamp of the next sent TSO message that is held by the RTI for delivery to the requesting federate, if there are any. There may not be any messages/events with the returned time available for the invoking federate.

Supplied Arguments

- None

Returned Arguments

- Minimum of
 - the invoking federate's LBTS and
 - the minimum time stamp of all sent TSO messages queued for the invoking federate (if any).

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate receives its minimum next event time.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Query LBTS*
- *Query Federate Time*

8.19 Modify Lookahead

The *Modify Lookahead* service shall request a change to the actual value of the federate's lookahead. The specified lookahead value shall be greater than or equal to zero. If the requested value is greater than or equal to the federate's actual lookahead, the change shall take effect immediately and the requested lookahead shall become the actual lookahead. If the requested value is less than the federate's actual lookahead, the change shall take effect gradually as the federate advances its logical time and the actual lookahead is initially unchanged. Specifically, the federate's actual lookahead shall decrease by T units each time logical time advances T units until the requested lookahead value is reached.

Supplied Arguments

- Requested value of lookahead

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- If the requested lookahead is greater than or equal to the federate's actual lookahead, the federate's actual lookahead shall be set to the requested value.
- If the requested lookahead is less than the federate's actual lookahead, the RTI shall be informed of the federate's requested lookahead value.

Exceptions

- The lookahead time is invalid.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Query Lookahead*

8.20 Query Lookahead

The *Query Lookahead* service shall query the RTI for the current value of the federate's actual lookahead. The current value of actual lookahead may differ temporarily from the requested lookahead given in the *Modify Lookahead* service if the federate is attempting to reduce its actual lookahead value.

Supplied Arguments

- None

Returned Arguments

- Federate's current value of actual lookahead

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate receives the current value of its actual lookahead.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Modify Lookahead*

8.21 Retract

The *Retract* service shall be used by a federate to notify the federation execution that a message/event previously sent by the federate is to be retracted. The *Update Attribute Values*, *Send Interaction*, and *Delete Object Instance* services shall return an event retraction designator that is used to specify the event that is to be retracted. Retracting an event shall cause the invocation of the *Request Retraction* † service in all the federates that received the original event.

Retracting a *Delete Object Instance* message shall result in the reconstitution of the corresponding object instance. This shall cause the ownership reassumption of the attributes of the affected object instance by the federates that owned them at the time of the *Delete Object Instance* service invocation.

Only messages sent in TSO may be retracted. A federate may not retract messages in its past. A message shall be in a federate's past if its time is earlier than the federate's current logical time.

Supplied Arguments

- Event retraction designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The federate has issued *Update Attribute Values*, *Send Interaction*, or *Delete Object Instance* service invocations previously and obtained the event retraction designators.
- The message associated with the specified retraction designator is not in the federate's past.

Post-conditions

- The RTI is informed that the federate requests to retract the specified event.

Exceptions

- The event retraction designator is invalid.
- The retraction designator is associated with a message in the federate's past.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Request Retraction* †

8.22 Request Retraction †

If the RTI receives a legal *Retract* service invocation for an event that has already been delivered to a federate, the *Request Retraction* † service shall be invoked on that federate. If the event in question has not been delivered to a federate, this service shall not be invoked on that federate; the event shall be removed from the RTI's event queue and never delivered to the federate.

Supplied Arguments

- Event retraction designator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The retracted event has been delivered to the federate.

Post-conditions

- The federate has been directed to retract the specified event.

Exceptions

- The event is not known.
- Federate internal error

Related Services

- *Retract*

8.23 Change Attribute Order Type

The preferred order type for each attribute of an object instance shall be initialized from the object class description in the FED. A federate may choose to change the preferred order type during execution. Invoking the *Change Attribute Order Type* service shall change the order type for all future *Update Attribute Values* service invocations for the specified instance attributes. When the ownership of an instance attribute is changed, the preferred order type shall revert to that defined in the FED.

Supplied Arguments

- Object instance designator
- Set of attribute designators
- Order type

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- An object instance with the specified designator exists.
- The specified class attributes are available attributes of the object instance's known class.
- The attributes are defined in the FED.
- The federate owns the instance attributes.

Post-conditions

- The order type is changed for the specified instance attributes.

Exceptions

- The object instance is not known.
- The specified class attributes are not available attributes of the known object class.
- The federate does not own the specified instance attributes.
- The order type is invalid.
- The federate is not a federate execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Update Attribute Values*
- *Change Attribute Transportation Type*

8.24 Change Interaction Order Type

The preferred order type of each interaction shall be initialized from the interaction class description in the FED. A federate may choose to change the preferred order type during execution. Invoking the *Change Interaction Order Type* service shall change the order type for all future *Send Interaction* and *Send Interaction with Region* service invocations for the specified interaction class for the invoking federate only.

Supplied Arguments

- Interaction class designator
- Order type

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The federate is publishing the interaction class.

Post-conditions

- The preferred order type is changed for the specified interaction class.

Exceptions

- The interaction class is not defined in FED.
- The federate is not publishing the interaction class.
- The order type is invalid.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Send Interaction*
- *Send Interaction with Region*
- *Change Interaction Transportation Type*

9. Data distribution management

9.1 Overview

Data distribution management (DDM) services shall be used by federates to reduce both the transmission and the reception of irrelevant data. Whereas declaration management services provide information on data relevance at the class attribute level, data distribution management services add the capability to further refine the data requirements at the instance attribute level. Producers of data shall employ DDM services to assert properties of their data in terms of user-defined spaces. Consumers of data shall employ DDM services to specify their data requirements in terms of the same spaces. The RTI distributes data from producers to consumers based on matches between these properties and requirements.

The DDM services shall be based on the following concepts and terms:

- A *dimension* shall be a named coordinate axis segment defined in the FED. The RTI shall provide a single *coordinate axis segment* defined by a pair of values. This provides a single basis for all dimensions defined in the FED. The first component of the pair shall be called *axis lower bound*, and the second component shall be called *axis upper bound*. All dimensions shall be based on the same coordinate-axis segment and thus shall have the same lower and upper bounds.
- A *routing space* shall be a named sequence of dimensions, which shall form a multi-dimensional coordinate system. Routing spaces shall be defined in the FED by indicating the dimensions that form the routing space. Routing spaces defined in the FED shall be said to be *available*. Additionally, the RTI shall provide an implicitly defined *default routing space*.
- A *range* shall be a continuous interval on a dimension defined by a pair of values. The first component of the pair shall be called *range lower bound*, and the second component shall be called *range upper bound*.
- An *extent* shall be a sequence of ranges, one for each dimension in the routing space.
- A *region* shall be a set of extents bound to a routing space. A region shall define a sub-space within the routing space. The RTI shall provide a *default region* for every routing space. The default region shall cover the entire routing space.

The following relationships, established in the FED, shall pertain to routing spaces:

- A class attribute shall be either explicitly *bound to* an available routing space or implicitly bound to the default routing space.
- An interaction class shall be either explicitly *bound to* an available routing space or implicitly bound to the default routing space.

The following relationship, established through DDM services, shall pertain to regions:

A region shall be *bound to* an available routing space by the *Create Region* service. This binding shall be revoked by the *Delete Region* service. Invoking the *Modify Region* service for a region shall notify the RTI about modifications to the extents of that region.

The following relationships, established through DDM services, shall pertain to object classes, class attributes, object instances, and instance attributes:

- A region shall be *used for update* of an instance attribute if the federate has used the instance attribute and region as arguments either
 - in the *Register Object Instance With Region* service or

— in the *Associate Region For Updates* service.

Subsequently invoking the *Unassociate Region For Update* service for the same (object instance, region) pair or the *Associate Region For Updates* service for the same (object instance, region) pair without providing the instance attribute shall cause the region not to be used for update of that instance attribute.

A region that is used for update of an instance attribute shall be a sub-space of the routing space to which the instance attribute corresponding class attribute is explicitly bound in the FED.

The default region shall be used for update of an instance attribute if no other region is used for update of that instance attribute.

A federate shall use a region for update to assert properties for the associated instance attribute when invoking *Update Attribute Values* service. If a federate loses ownership of an instance attribute that is associated with a region, that instance attribute shall no longer be associated with the corresponding region.

- A region shall be *used for subscription* of a class attribute if the federate has used the class attribute and region as arguments in the *Subscribe Object Class Attributes With Region* service for the (object class, region) pair. Subsequently invoking the *Unsubscribe Object Class With Region* service for the same (object class, region) pair or invoking the *Subscribe Object Class Attributes With Region* service for the same (object class, region) pair without providing the class attribute shall cause the region not to be used for subscription of that class attribute.

A region that is used for subscription of a class attribute shall be a sub-space of the routing space to which the class attribute is explicitly bound in the FED.

The default region shall be used for subscription of a class attribute if the federate has used the class attribute as an argument in the *Subscribe Object Class Attributes* service. Subsequently invoking the *Unsubscribe Object Class* service for the same object class or invoking *Subscribe Object Class Attributes* service for the same object class without providing the class attribute shall cause the default region not to be used for subscription of that class attribute.

A federate shall use a region for subscription to specify requirements for reflecting attribute values.

The following relationships, established through DDM services, shall pertain to interaction classes, parameters, and interactions:

- A region shall be *used for sending* an interaction during the invocation of the *Send Interaction With Region* service.

A region that is used for sending an interaction shall be a sub-space of the routing space to which the corresponding interaction class is explicitly bound in the FED.

The default region shall be used for sending an interaction during the invocation of the *Send Interaction* service.

A federate shall use a region for sending to assert properties for the interaction in the *Send Interaction* service.

- A region shall be *used for subscription* of an interaction class if the federate has used the interaction class and region as arguments in the *Subscribe Interaction Class With Region* service

for the region. Subsequently invoking the *Unsubscribe Object Class With Region* service for the same region shall cause the region not to be used for subscription of that interaction class.

A region that is used for subscription of an interaction class shall be a sub-space of the routing space to which the interaction class is explicitly bound in the FED.

The default region shall be used for subscription of an interaction class if the federate has used the interaction class as an argument in the *Subscribe Interaction Class* service. Subsequently invoking the *Unsubscribe Object Class* service for the same interaction class shall cause the default region not to be used for subscription of that interaction class.

A federate shall use a region for subscription to establish requirements for receiving the associated interaction class.

A region used for update of instance attributes or for sending interactions shall be called an *update region*.

A region used for subscription of either class attributes or interaction classes shall be called a *subscription region*.

An update region and a subscription region shall *overlap* if and only if the regions are bound to the same routing space and the corresponding extent sets overlap. Two extent sets shall overlap if there is an extent in each set, such that the two extents overlap. Two extents shall overlap if all their ranges overlap pairwise. Two ranges $A = [a_{\text{lower}}, a_{\text{upper}})$ and $B = [b_{\text{lower}}, b_{\text{upper}})$ shall overlap, if and only if either $a_{\text{lower}} = b_{\text{lower}}$ or $(a_{\text{lower}} < b_{\text{upper}}$ and $b_{\text{lower}} < a_{\text{upper}})$.

Routing spaces, regions, and the usage for update or subscription by federates shall determine to which federates the RTI distributes data. Whenever an update region in either the owning federate (for invocations of the *Update Attribute Values* service) or the sending federate (for invocations of the *Send Interaction* service) and a subscription region in the subscribing federate overlap, the RTI shall ensure that the updated instance attribute values are reflected and the sent interactions are received. Figure 13 is an example of a routing space with two dimensions.

The usage of dimensions for the specification of data distribution properties and requirements shall be left to the federation. The effects of DDM services shall be independent of federation time.

PLACE

Figure 13—Routing space of two dimensions

HERE

9.2 Create Region

The *Create Region* service shall create a region that has the dimensions of the specified routing space and the specified number of extents. The extent set shall delineate the region within the routing space. The region may be used for either update or subscription.

Supplied Arguments

- Routing space designator
- Extents

Returned Arguments

- Region

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The routing space is defined in the FED.

Post-conditions

- A region exists that has the dimensions of the specified routing space.

Exceptions

- The routing space is not defined in the FED.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Register Object Instance With Region*
- *Associate Region For Updates*
- *Subscribe Object Class Attributes With Region*
- *Subscribe Interaction Class With Region*
- *Send Interaction With Region*
- *Modify Region*
- *Delete Region*

9.3 Modify Region

The *Modify Region* service shall inform the RTI about changes to the extent set of the region.

Supplied Arguments

- Region
- Extents

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The region exists.

Post-conditions

- The region has a new delineation.

Exceptions

- The region is not known.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Create Region*

9.4 Delete Region

The *Delete Region* service shall delete the specified region. A region in use for subscription or update shall not be deleted.

Supplied Arguments

- Region

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The region exists.
- The region is not in use.

Post-conditions

- The region no longer exists.

Exceptions

- The region is not known.
- The region is in use.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Create Region*

9.5 Register Object Instance With Region

The *Register Object Instance With Region* service shall create a unique object instance designator and link it with an object instance of the supplied class. All instance attributes whose corresponding class attributes are currently published by the registering federate shall be set as owned by the registering federate.

This service shall be used to create an object instance and simultaneously associate update regions with attributes of that object instance. This service shall be an atomic operation that can be used in place of *Register Object Instance* followed by *Associate Region For Updates*. Those instance attributes whose corresponding class attributes are currently published but are not supplied in the service invocation shall be associated with the default regions in the routing spaces to which the class attributes are bound.

If the optional object instance name argument is supplied, that name shall be associated with the object instance.

Supplied Arguments

- Object class designator
- Set of attribute designator/region pairs
- Optional object instance name

Returned Arguments

- Object instance designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The federate is publishing the object class.
- The class attributes are available.
- The federate is publishing the specified class attributes of the specified object class.
- The regions exist.
- For each class attribute/region pair, the routing space denoted by the region is the routing space bound to the class attribute in the FED.
- If the optional object instance name argument is supplied, that name is unique.

Post-conditions

- The returned object instance designator is associated with the object instance.
- The federate owns the instance attributes that are currently published for the specified object class.
- The specified instance attributes are associated with the respective regions for future *Update Attribute Values* service invocations.
- If the optional object instance name argument is supplied, that name is associated with the object instance.

Exceptions

- The object class is not defined in FED.
- The federate is not publishing the object class.
- The class attribute is not available at the known class of the object instance.
- The federate is not publishing the class attribute.
- The region is not known.

- The routing space denoted by region is not the one bound to the class attribute in the FED.
- The object instance name is not unique.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Publish Object Class*
- *Register Object Instance*
- *Create Region*
- *Discover Object †*
- *Attribute Ownership Acquisition Notification †*

9.6 Associate Region For Updates

The *Associate Region For Updates* service shall associate a region to be used for updates with instance attributes of a specific object instance.

Associating a region with an instance attribute shall mean that the federate shall ensure that the properties of the instance attribute fall within the extents of the associated region at the time when an *Update Attribute Values* service is invoked.

The association shall be used by the *Update Attribute Values* service to route data to subscribers whose subscription regions overlap the specified update region. Based on the object instance and the specified region, this service shall perform

- An addition to the group of associations if the object instance/region pair had no attribute set linked with it, or
- A replacement in the group of associations if there is an attribute set currently linked with the object instance/region pair.

The *Unassociate Region For Updates* service shall be used to remove an established association from the group of associations.

Those instance attributes that are implicitly unassociated by the invocation shall be associated with the default region.

Supplied Arguments

- Object instance designator
- Set of attribute designator/region pairs

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object instance exists.
- The class attributes are available.
- The federate owns the specified instance attributes.
- The regions exist.
- For each class attribute/region pair, the routing space denoted by the region is the routing space bound to the class attribute in the FED.

Post-conditions

- The specified instance attributes are associated with the respective regions for future invocations of the *Update Attribute Values* service.

Exceptions

- The object instance is not known.
- The class attribute is not available.
- The federate does not own the instance attribute.
- The region is not known.
- The routing space denoted by region is not the one bound to the class attribute in the FED.
- The federate is not a federation execution member.

- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Create Region*
- *Modify Region*
- *Update Attribute Values*
- *Unassociate Region For Updates*

9.7 Unassociate Region For Updates

The *Unassociate Region For Updates* service shall remove the association between the region and all instance attributes associated with that region.

The instance attributes that are unassociated by the invocation shall be associated with the default region.

Supplied Arguments

- Object instance designator
- Region

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object instance exists.
- The region is associated with attributes of the object instance.

Post-conditions

- The region is no longer associated with any attributes of the object instance.

Exceptions

- The object instance is not known.
- The region was not associated with attributes of the object instance.
- The region is not known.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Associate Region For Updates*
- *Create Region*
- *Update Attribute Values*
- *Register Object Instance With Region*

9.8 Subscribe Object Class Attributes With Region

The *Subscribe Object Class Attributes With Region* service shall specify an object class for which the RTI is to begin notifying the federate of discovery of instantiated object instances when at least one of that instance's attributes are in scope. This service and subsequent related RTI operations shall behave similarly to *Subscribe Object Class Attributes* and subsequent related RTI operations as described in 5.6. This service shall provide additional functionality in that the intersection of the relevant subscription and update regions affect the subsequent RTI operations, as described in the beginning of this clause.

There shall be only one attribute set linked with each object class/region pair in the group of subscriptions that is defined to support data distribution management. Based on each of the implied object class/region pairs, this service shall perform one of the following actions with the specified attribute set:

- An addition to the group of subscriptions if the object class/region pair has no attribute set linked with it, or
- A replacement in the group of subscriptions if there is currently an attribute set linked with the object class/region pair.

Invoking this service with an empty set of attributes shall be equivalent to invoking the *Unsubscribe Object Class With Region* service with the relevant object class.

If the optional passive subscription indicator indicates that this is a passive subscription,

- a) the invocation of this service shall not cause the *Start Registration For Object Class* † service or the *Turn Updates On For Object Instance* † service to be invoked at any other federate and
- b) if this invocation replaces a previous subscription that was active rather than passive, invocation of this service may cause the *Stop Registration for Object Class* † service or the *Turn Updates Off For Object Instance* † service to be invoked at one or more other federates.

If the optional passive subscription indicator is not present or indicates that this is an active subscription,

- a) the invocation of this service may cause the *Start Registration For Object Class* † service or the *Turn Updates On For Object Instance* † service to be invoked at one or more other federates and
- b) if this invocation replaces a previous subscription that was active rather than passive, invocation of this service may cause the *Turn Updates Off For Object Instance* † service to be invoked at one or more other federates.

Supplied Arguments

- Object class designator
- Set of attribute designator/region pairs
- Optional passive subscription indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The class attributes are available.
- The regions exist.
- For each class attribute/region pair, the routing space denoted by the region is the routing space bound to the class attribute in the FED.

Post-conditions

- The RTI has been informed of the federate's requested subscription.

Exceptions

- The object class is not defined in the FED.
- The class attribute is not available.
- The region is not known.
- The routing space denoted by region is not the one bound to the class attribute in the FED.
- Invalid passive subscription indicator.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Unsubscribe Object Class With Region*
- *Publish Object Class*
- *Discover Object †*
- *Attributes In Scope †*
- *Reflect Attribute Values †*
- *Create Region*
- *Start Registration For Object Class †*
- *Stop Registration For Object Class †*
- *Turn Updates On For Object Instance †*
- *Turn Updates Off For Object Instance †*

9.9 Unsubscribe Object Class With Region

The Unsubscribe Object Class With Region service shall inform the RTI that it shall stop notifying the federate of object instance discoveries for the specified object class. The unsubscribe shall be confined to all subscriptions using the specified region.

Supplied Arguments

- Object class designator
- Region

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object class is defined in the FED.
- The federate is subscribed to the object class for the region.
- The region exists.

Post-conditions

- The RTI has been informed of the federate's requested unsubscription.

Exceptions

- The object class is not defined in the FED.
- The region is not known.
- The federate is not subscribed to the object class for the region.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Subscribe Object Class Attributes With Region*

9.10 Subscribe Interaction Class With Region

The Subscribe Interaction Class With Region service shall specify the class of interactions that should be delivered to the federate, taking the region into account. This service and subsequent related RTI operations shall behave similarly to *Subscribe Interaction Class* and subsequent related RTI operations as described in 5.8. This service shall provide additional functionality in that the intersection of the relevant subscription and interaction send regions affect the subsequent RTI operations, as described in the beginning of this clause.

If the specified region is currently in the group of regions associated with the specified interaction class subscription, this service is a no-op; otherwise an addition to the group shall be performed.

If the optional passive subscription indicator indicates that this is a passive subscription, the invocation of this service shall not cause the *Turn Interactions On* \nrightarrow service to be invoked at any other federate and

If the optional passive subscription indicator is not present or indicates that this is an active subscription, the invocation of this service may cause the *Turn Interactions On* \nrightarrow service to be invoked at one or more other federates.

Supplied Arguments

- Interaction class designator
- Region
- Optional passive subscription indicator

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The region exists.
- The routing space denoted by the region is the routing space bound to the interaction class in the FED.

Post-conditions

- The RTI has been informed of the federate's requested subscription.

Exceptions

- The interaction class is not defined in the FED.
- The region is not known.
- The routing space denoted by region is not the one bound to the interaction class in the FED.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Subscribe Interaction Class*
- *Unsubscribe Interaction Class with Region*
- *Publish Interaction Class*

- *Receive Interaction* †
- *Create Region*
- *Turn Interactions On* †
- *Turn Interactions Off* †

9.11 Unsubscribe Interaction Class With Region

The *Unsubscribe Interaction Class With Region* service shall inform the RTI no longer to notify the federate of sent interactions of the specified class that are in the specified region.

Supplied Arguments

- Interaction class designator
- Region

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The federate is subscribed to the interaction class for the region.
- The region exists.

Post-conditions

- The RTI has been informed of the federate's requested unsubscription.

Exceptions

- The interaction class is not defined in the FED.
- The region is not known.
- The federate is not subscribed to the interaction class for the region.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Subscribe Interaction Class with Region*

9.12 Send Interaction With Region

The *Send Interaction With Region* service shall send an interaction into the federation. The interaction parameters may be those in the specified class and all super-classes, as defined in the FED. The region shall be used to limit the scope of potential receivers of the interaction. The service shall return a federation-unique event retraction designator. An event retraction designator shall be returned only if the federation time argument is supplied.

Supplied Arguments

- Interaction class designator
- Set of parameter-designator/value pairs
- User-supplied tag
- Region
- Optional federation time

Returned Arguments

- Optional event retraction designator

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The interaction class is defined in the FED.
- The federate is publishing the interaction class.
- The interaction parameters are available.
- The region exists.
- The routing space denoted by the region is the routing space bound to the interaction class in the FED.

Post-conditions

- The RTI has received the interaction.

Exceptions

- The federate is not publishing the specified interaction class.
- The interaction class is not defined in FED.
- The interaction parameter is not available.
- The federation time is invalid (if optional time argument is supplied).
- The region is not known.
- The routing space denoted by region is not the one bound to the interaction class in the FED.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Time Advance Request*
- *Next Event Request*
- *Time Advance Grant* †
- *Receive Interaction* †
- *Publish Interaction Class*

- *Retract*
- *Create Region*

9.13 Request Attribute Value Update With Region

The *Request Attribute Value Update With Region* service shall be used to stimulate the update of values of specified attributes. When this service is used, the RTI shall solicit the current values of the specified attributes from their owners using the *Provide Attribute Value Update †* service. When an object class is specified, the RTI shall solicit the specified attributes for all the object instances of that class. When an object instance designator is specified, the RTI shall solicit the specified attributes for the particular object. The resulting *Provide Attribute Value Update †* service invocations issued by the RTI shall be consistent with the region arguments to this service. An invocation shall be consistent with the region arguments if the instance attributes in an updating federate are associated with a region that overlaps the corresponding region specified as an argument to this service. The federation time of any resulting *Reflect Attribute Values †* service invocations shall be determined by the updating federate.

Supplied Arguments

- Object instance designator or object class designator
- Set of attribute designator/region pairs

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The object instance exists (when first argument is an object instance designator).
- The object class is defined in the FED (when first argument is an object class designator).
- The class attributes are available.
- The regions exist.
- For each class attribute/region pair, the routing space denoted by the region is the routing space bound to the class attribute in the FED.

Post-conditions

- The request for the updated attribute values has been received by the RTI.

Exceptions

- The object is not known.
- The object class is not defined in the FED.
- The class attribute is not available.
- The region is not known.
- The routing space denoted by region is not the one bound to the class attribute in the FED.
- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Provide Attribute Value Update †*
- *Update Attribute Values*
- *Create Region*

9.14 Change Thresholds †

The *Change Thresholds* † service shall provide the federate with the new values of the thresholds for each of the dimensions of a routing space.

Supplied Arguments

- Region
- Set of thresholds

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.
- The region exists.

Post-conditions

- The federate is informed of the new threshold values for the region.

Exceptions

- The region is not known.
- Federate internal error

Related Services

- *Create Region*
- *Modify Region*

10. Support services

10.1 Overview

This clause describes miscellaneous services utilized by federates for performing such actions as

- Name-to-handle and handle-to-name transformation
- Setting advisory switches

All class name arguments shall be completely specified, including all super-class names.

10.2 Get Object Class Handle

The *Get Object Class Handle* service shall return the object class handle associated with the supplied object class name.

Supplied Arguments

- Object class name

Returned Arguments

- Object class handle

Pre-conditions

- The specified object class is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested object class handle.

Exceptions

- The object class is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Object Class Name*

10.3 Get Object Class Name

The *Get Object Class Name* service shall return the object class name associated with the supplied object class handle.

Supplied Arguments

- Object class handle

Returned Arguments

- Object class name

Pre-conditions

- The specified object class is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested object class name.

Exceptions

- The object class is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Object Class Handle*

10.4 Get Attribute Handle

The *Get Attribute Handle* service shall return the attribute handle associated with the supplied attribute name and object class.

Supplied Arguments

- Attribute name
- Object class handle

Returned Arguments

- Attribute handle

Pre-conditions

- The specified object class is defined in the FED.
- The specified class attribute is an available attribute of the specified object class.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested attribute handle.

Exceptions

- The object class is not defined in the FED.
- The specified object class attribute is not an available attribute of the specified object class.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Attribute Name*

10.5 Get Attribute Name

The *Get Attribute Name* service shall return the attribute name associated with the supplied attribute handle and object class.

Supplied Arguments

- Attribute handle
- Object class handle

Returned Arguments

- Attribute name

Pre-conditions

- The specified object class is defined in the FED.
- The specified class attribute is an available attribute of the specified object class.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested attribute name

Exceptions

- The object class is not defined in the FED.
- The specified object class attribute is not an available attribute of the specified object class.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Attribute Handle*

10.6 Get Interaction Class Handle

The *Get Interaction Class Handle* service shall return the interaction class handle associated with the supplied interaction class name.

Supplied Arguments

- Interaction class name

Returned Arguments

- Interaction class handle

Pre-conditions

- The specified interaction class is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested interaction class handle.

Exceptions

- The interaction class is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Interaction Class Name*

10.7 Get Interaction Class Name

The *Get Interaction Class Name* service shall return the interaction class name associated with the supplied interaction class handle.

Supplied Arguments

- Interaction class handle

Returned Arguments

- Interaction class name

Pre-conditions

- The specified interaction class is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested interaction class name.

Exceptions

- The interaction class is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Interaction Class Handle*

10.8 Get Parameter Handle

The *Get Parameter Handle* service shall return the parameter handle associated with the supplied parameter name and interaction class.

Supplied Arguments

- Parameter name
- Interaction class handle

Returned Arguments

- Parameter handle

Pre-conditions

- The specified interaction class is defined in the FED.
- The specified parameter is an available parameter of the specified interaction class.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested parameter handle.

Exceptions

- The interaction class is not defined in the FED.
- The parameter is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Parameter Name*

10.9 Get Parameter Name

The *Get Parameter Name* service shall return the parameter name associated with the supplied parameter handle and interaction class.

Supplied Arguments

- Parameter handle
- Interaction class handle

Returned Arguments

- Parameter name

Pre-conditions

- The specified interaction class is defined in the FED.
- The specified parameter is an available parameter of the specified interaction class.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested parameter name.

Exceptions

- The interaction class is not defined in the FED.
- The parameter is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Parameter Handle*

10.10 Get Object Instance Handle

The *Get Object Instance Handle* service shall return the handle of the object instance with the supplied name.

Supplied Arguments

- Object instance name

Returned Arguments

- Object instance handle

Pre-conditions

- The object instance with the specified name exists.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested object instance handle.

Exceptions

- The object instance is not known.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Object Instance Name*

10.11 Get Object Instance Name

The *Get Object Instance Name* service shall return the name of the object instance with the supplied handle.

Supplied Arguments

- Object instance handle

Returned Arguments

- Object instance name

Pre-conditions

- The object instance with the specified name exists.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested object instance name.

Exceptions

- The object instance is not known.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Object Instance Handle*

10.12 Get Routing Space Handle

The *Get Routing Space Handle* service shall return the routing space handle associated with the supplied routing space name.

Supplied Arguments

- Routing space name

Returned Arguments

- Routing space handle

Pre-conditions

- The specified routing space is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested routing space handle.

Exceptions

- The routing space is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Routing Space Name*

10.13 Get Routing Space Name

The *Get Routing Space Name* service shall return the routing space name associated with the supplied routing space handle.

Supplied Arguments

- Routing space handle

Returned Arguments

- Routing space name

Pre-conditions

- The specified routing space is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested routing space name.

Exceptions

- The routing space is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Routing Space Handle*

10.14 Get Dimension Handle

The *Get Dimension Handle* service shall return the dimension handle associated with the supplied dimension name and routing space.

Supplied Arguments

- Dimension name
- Routing space handle

Returned Arguments

- Dimension handle

Pre-conditions

- The specified routing space is defined in the FED.
- The specified dimension is defined in the specified routing space in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested dimension handle.

Exceptions

- The routing space is not defined in the FED.
- The dimension is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Dimension Name*

10.15 Get Dimension Name

The *Get Dimension Name* service shall return the dimension name associated with the supplied dimension handle and routing space.

Supplied Arguments

- Dimension handle
- Routing space handle

Returned Arguments

- Dimension name

Pre-conditions

- The specified routing space is defined in the FED.
- The specified dimension is defined in the specified routing space in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested dimension name.

Exceptions

- The routing space is not defined in the FED.
- The dimension is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Dimension Handle*

10.16 Get Attribute Routing Space Handle

The *Get Attribute Routing Space Handle* service shall return the routing space associated with the supplied attribute and object class.

Supplied Arguments

- Attribute handle
- Object class handle

Returned Arguments

- Routing space handle

Pre-conditions

- The specified object class is defined in the FED.
- The specified class attribute is an available attribute of the specified object class.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested routing space handle.

Exceptions

- The object class is not defined in the FED.
- The specified object class attribute is not an available attribute of the specified object class.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- None

10.17 Get Object Class

The *Get Object Class* service shall return the known object class of the supplied object instance.

Supplied Arguments

- Object instance handle

Returned Arguments

- Object class handle

Pre-conditions

- The specified object instance exists.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the known object class of the specified object instance.

Exceptions

- The object instance is not known.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- None

10.18 Get Interaction Routing Space Handle

The *Get Interaction Routing Space Handle* service shall return the routing space associated with the supplied interaction class.

Supplied Arguments

- Interaction class handle

Returned Arguments

- Routing space handle

Pre-conditions

- The specified interaction class is defined in the FED.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested routing space handle.

Exceptions

- The interaction is not defined in the FED.
- The federate is not a federation execution member.
- RTI internal error

Related Services

- None

10.19 Get Transportation Handle

The *Get Transportation Handle* service shall return the transportation handle associated with the supplied transportation name.

Supplied Arguments

- Transportation name

Returned Arguments

- Transportation handle

Pre-conditions

- The transportation name is defined.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested transportation handle.

Exceptions

- Name not found
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Transportation Name*

10.20 Get Transportation Name

The *Get Transportation Name* service shall return the transportation name associated with the supplied transportation handle.

Supplied Arguments

- Transportation handle

Returned Arguments

- Transportation name

Pre-conditions

- The transportation handle is defined.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested transportation name.

Exceptions

- Invalid transportation handle
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Transportation Handle*

10.21 Get Ordering Handle

The *Get Ordering Handle* service shall return the ordering handle associated with the supplied ordering name.

Supplied Arguments

- Ordering name

Returned Arguments

- Ordering handle

Pre-conditions

- The ordering name is defined.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested ordering handle.

Exceptions

- Name not found
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Ordering Name*

10.22 Get Ordering Name

The *Get Ordering Name* service shall return the ordering name associated with the supplied ordering handle.

Supplied Arguments

- Ordering handle

Returned Arguments

- Ordering name

Pre-conditions

- The ordering handle is defined.
- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The federate has the requested ordering name.

Exceptions

- Invalid ordering handle
- The federate is not a federation execution member.
- RTI internal error

Related Services

- *Get Ordering Handle*

10.23 Enable Class Relevance Advisory Switch

The *Enable Class Relevance Advisory Switch* service shall set the Class Relevance Advisory switch on.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Class Relevance Advisory switch is turned on.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Disable Class Relevance Advisory Switch*
- *Start Registration For Object Class* †
- *Stop Registration For Object Class* †

10.24 Disable Class Relevance Advisory Switch

The *Disable Class Relevance Advisory Switch* service shall set the Class Relevance Advisory Switch off.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Class Relevance Advisory switch is turned off.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Class Relevance Advisory Switch*
- *Start Registration For Object Class* †
- *Stop Registration For Object Class* †

10.25 Enable Attribute Relevance Advisory Switch

The *Enable Attribute Relevance Advisory Switch* service shall set the Attribute Relevance Advisory switch on.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Attribute Relevance Advisory switch is turned on.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Disable Attribute Relevance Advisory Switch*
- *Turn Updates On For Object Instance* †
- *Turn Updates Off For Object Instance* †

10.26 Disable Attribute Relevance Advisory Switch

The *Disable Attribute Relevance Advisory Switch* service shall set the Attribute Relevance Advisory switch off.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Attribute Relevance Advisory switch is turned off.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Attribute Relevance Advisory Switch*
- *Turn Updates On For Object Instance* †
- *Turn Updates Off For Object Instance* †

10.27 Enable Attribute Scope Advisory Switch

The *Enable Attribute Scope Advisory Switch* service shall set the Attribute Scope Advisory switch on.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Attribute Scope Advisory switch is turned on.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Disable Attribute Scope Advisory Switch*
- *Attributes In Scope* †
- *Attributes Out Of Scope* †

10.28 Disable Attribute Scope Advisory Switch

The *Disable Attribute Scope Advisory Switch* service shall set the Attribute Scope Advisory switch off.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Attribute Scope Advisory switch is turned off.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Attribute Scope Advisory Switch*
- *Attributes In Scope* †
- *Attributes Out Of Scope* †

10.29 Enable Interaction Relevance Advisory Switch

The *Enable Interaction Relevance Advisory Switch* service shall set the Interaction Relevance Advisory switch on.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Interaction Relevance Advisory switch is turned on.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Disable Interaction Relevance Advisory Switch*
- *Tune Interactions On* †
- *Tune Interactions Off* †

10.30 Disable Interaction Relevance Advisory Switch

The *Disable Interaction Relevance Advisory Switch* service shall set the Interaction Relevance Advisory switch off.

Supplied Arguments

- None

Returned Arguments

- None

Pre-conditions

- The federation execution exists.
- The federate is joined to that federation execution.

Post-conditions

- The Interaction Relevance Advisory switch is turned off.

Exceptions

- The federate is not a federation execution member.
- Save in progress
- Restore in progress
- RTI internal error

Related Services

- *Enable Interaction Relevance Advisory Switch*
- *Tune Interactions On* †
- *Tune Interactions Off* †

11. Management object model (MOM)

Management object model (MOM) facilities shall be used by federates and the RTI to provide insight into the operations of federates and the RTI and to control the functioning of the RTI, the federation, and individual federates. The ability to monitor and control elements of a federation shall be required for proper functioning of a federation execution.

MOM shall satisfy these requirements by utilizing predefined HLA constructs: objects and interactions. In effect, the RTI shall publish object classes and shall register and update values of attributes of object instances; shall subscribe to and receive some interaction classes; and shall publish and send other interaction classes. A federate charged with controlling a federation execution shall subscribe to the object classes, reflect the updates, publish and send some interaction classes, and subscribe to and receive other interaction classes.

The MOM object class structure is depicted in Figure 14.

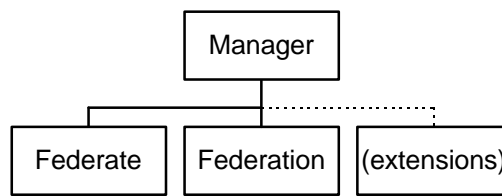


Figure 14—MOM object class structure

- Object class *Manager.Federate* shall contain attributes that describe the state of a federate. The RTI shall publish the class and register one object instance of this class for each federate in the federation. The RTI shall update the information periodically, based on timing data provided in *Manager.Federate.Adjust* interactions. Information shall be contained in an object instance that includes identifying information about the federate, measures of the federate's time state, and the status of queues maintained by the RTI for the federate.
- Object class *Manager.Federation* shall contain attributes that describe the state of the federation execution. The RTI shall publish the class and register one object instance of this class for the federation.

The MOM interaction class structure is depicted in Figure 15.

- Interaction classes that are subclasses of *Manager.Federate.Adjust* shall be acted upon by the RTI. They shall permit a managing federate to adjust the way the RTI performs when responding to another federate and how it shall respond and report to the managing federate.

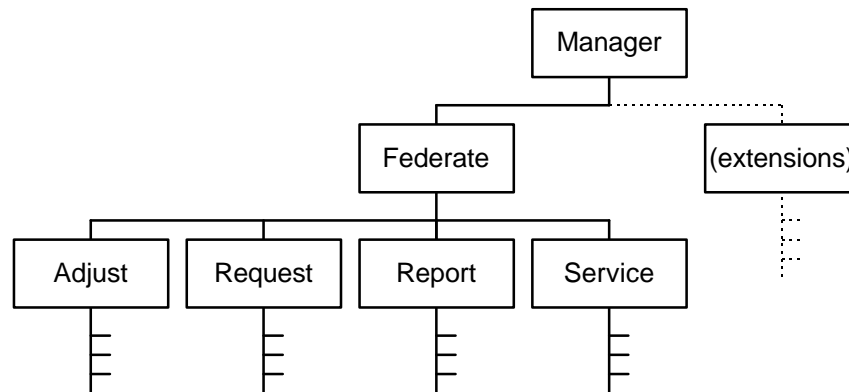


Figure 15—MOM interaction class structure

- Interaction classes that are subclasses of *Manager.Federate.Request* shall be acted upon by the RTI. They shall cause the RTI to send subclasses of *Manager.Federate.Report* interaction class.
- Interaction classes that are subclasses of *Manager.Federate.Report* shall be sent by the RTI. They shall respond to interaction classes that are subclasses of *Manager.Federate.Request* class interactions. They shall describe some aspect of the federate such as its object class subscription tree.
- Interaction classes that are subclasses of *Manager.Federate.Service* shall be acted upon by the RTI. They shall invoke RTI services on behalf of another federate. For services that are normally invoked by a federate, they shall cause the RTI to react as if the service was invoked by the federate (for example, a managing federate could change the time-regulating state of another federate). Services that are normally callbacks from the RTI to a federate shall cause the RTI to invoke the callback.

All MOM object classes, interaction classes, attributes, and parameters shall be predefined in the FED file. These definitions may not be revised.

MOM definitions may be extended, however; they may be augmented with additional subclasses, class attributes, or parameters. These new elements shall not be acted upon directly by the RTI—they shall be acted upon by federates in the federation.

The MOM object classes may be extended by adding subclasses or class attributes. Without extensions, the RTI shall publish *Manager.Federate* and *Manager.Federation* classes with predefined MOM class attributes, register an instance, and update the values of the predefined instance attributes. The RTI shall not subscribe to any object class. Valid methods for extending the MOM object classes shall be

- Subclasses may be added to any MOM object class. Here, the federate shall publish the object class and its attributes, register an instance of the new class, and update values of instance attributes of the object instance according to dictates of the federation execution. Note that the instance of the subclass shall be separate from the MOM object instance that is registered by the RTI. Therefore, instance attributes that are inherited by the extension subclass from the MOM predefined class shall not be updated by the RTI.
- Attributes may be added to any MOM object class. Here, the federate shall publish the object class with the new class attributes; and shall subscribe to the object class and attributes in it, discover and reflect updates to learn the object instance in question, and update the values of the new instance attributes using the discovered object instance designator. Note that the instance that the federate shall update with the new instance attributes shall be the same as the MOM object instance that is

registered by the RTI.

The MOM interaction classes may be extended by adding subclasses or parameters. There shall be three categories of extension of MOM interaction classes:

- a) Classes of interaction that the RTI shall send (subclasses of *Manager.Federate.Report*). The RTI shall publish at the MOM leaf-class level (e.g., *Manager.Federate.Report.Alert*). It shall send interactions containing all predefined parameters for that interaction class. Valid methods for extending this type of MOM interaction class shall be
 - Subclasses may be added to these MOM interaction classes. The RTI shall not send interactions of these subclasses. If federates subscribe to the subclass, they shall receive the full interaction. If they subscribe to the class of which the extension is a subclass, the interaction shall be promoted to the subscribed class and any new parameters shall be lost.
 - Parameters may be added to any MOM interaction class. Interactions of these classes that are sent by the RTI shall not contain the new parameters.
- b) Classes of interaction that the RTI shall receive (subclasses of *Manager.Federate.Adjust*, *Manager.Federate.Request*, and *Manager.Federate.Service*). The RTI shall subscribe at the MOM leaf-class level (e.g., *Manager.Federate.Adjust.SetTiming*). It shall receive these interactions and process all predefined parameters for that interaction class. Valid methods for extending this type of MOM interaction class shall be
 - Subclasses may be added to any MOM interaction class. If a federate sends an interaction of this class, the RTI shall receive a promoted version that shall contain only the parameters of the predefined interaction class.
 - Parameters may be added to any MOM interaction class. If a federate sends an interaction with extra parameters, the RTI shall receive the new parameters but shall ignore them and process only the predefined parameters.
- c) Classes of interaction that shall be neither sent nor received by the RTI. These classes of interaction shall be ignored by the RTI and may be formed in any way that is consistent with FOM development.

11.1 MOM objects

The MOM shall contain two predefined object classes: *Manager.Federate* and *Manager.Federation*, and the attributes associated with them.

The object classes are described in the following paragraphs. No instance attributes of these classes shall be transferable; the RTI shall never release ownership of the instance attributes.

NOTE—the data type of all instance attributes shall be text; the tables defining the attributes shall present a more specific data type—this shall represent the data type from which the text data shall be translated. For enumerated attributes, the values are presented in the form that shall be provided in the RTI API; specific values shall depend on the computer-language version of the API.

11.1.1 Object class *Manager.Federation*

The object class *Manager.Federation* shall contain RTI state variables relating to a federation execution. The RTI shall publish object class *Manager.Federation* and shall register one object instance for the federation execution. It shall not automatically update the values of the instance attributes; a federate shall use a *Request Attribute Value Update* service to obtain values for the instance attributes.

Table 4—Object class *Manager.Federation*

Attribute	Type	Description
-----------	------	-------------

Attribute	Type	Description
FederationName	string	Name of the federation to which the federate belongs
FederatesInFederation	handle list	Comma-separated list of the designators of federates that have joined the federation execution (null string if none)
RTIversion	string	Version of the RTI software
FEDid	string	Identifier associated with the FED data used by the federation
LastSaveName	string	Name associated with the last federation state save (null if no saves have occurred)
LastSaveTime	time	Logical time at which the last federation state save occurred (zero if no saves have occurred)
NextSaveName	string	Name associated with the next federation state save (null if no saves are scheduled)
NextSaveTime	time	Logical time at which the next federation state save is scheduled (zero if no saves are scheduled)

11.1.2 Object class *Manager.Federate*

This object class *Manager.Federate* shall contain RTI state variables relating to a federate. The RTI shall publish object class *Manager.Federate* and shall register one object instance for each federate in a federation. Dynamic attributes that shall be contained in an object instance shall be updated periodically, where the period should be determined by an interaction of the class *Manager.Federate.Adjust.SetTiming*. If this value is never set or is set to zero, no periodic update shall be performed by the RTI.

Table 5—Object class *Manager.Federate*

Attribute	Type	Description
Federate	handle	Designator of the federate returned by a join <i>FederationExecution</i> service invocation
FederateType	string	Type of the federate specified by the federate when it joined the federation
FederateName	string	Name of the federate specified by the federate when it joined the federation
FederateHost	string	Host name of the computer on which the federate is executing
RTIversion	string	Version of the RTI software being used
FEDid	string	Identifier associated with the FED data used by the federate
TimeConstrained	boolean	Whether the time advance of the federate is constrained by other federates
TimeRegulating	boolean	Whether the federate influences the time advance of other federates

Attribute	Type	Description
AsynchronousDelivery	boolean	Whether the RTI shall deliver receive-order messages to the federate while the federate's time manager state is "Idle" (only valid if the federate is time-constrained)
FederateState	enumerated	State of the federate; valid values are — Running — Save pending — Saving — Restore pending — Restoring
TimeManagerState	enumerated	State of the federate's time manager state; valid values are — Idle — Advance pending
FederateTime	time	Logical time of the federate (zero if logical time is not used)
Lookahead	time	Minimum duration into the future that a TSO event will be scheduled (zero if logical time is not used)
LBTS	time	Logical time of the LTBS (zero if logical time is not used)
MinNextEventTime	time	Minimum of the LBTS and the head of the TSO queue (zero if logical time is not used)
ROlength	long	Number of events stored in the RO queue
TSOlength	long	Number of events stored in the TSO queue
ReflectionsReceived	long	Total number of reflections received by the federate
UpdatesSent	long	Total number of updates sent by the federate
InteractionsReceived	long	Total number of interactions received by the federate
InteractionsSent	long	Total number of interactions sent by the federate
ObjectsOwned	long	Total number of object instances whose <i>PrivilegeToDelete</i> attribute is owned by the federate
ObjectsUpdated	long	Total number of object instances for which the federate updates at least one attribute value
ObjectsReflected	long	Total number of object instances for which the federate reflects updates of at least one attribute

11.2 MOM interactions

The MOM shall contain a single predefined interaction class, *Manager*, and a single subclass of that class, *Federate*. Subordinate to that level shall be four subclasses: *Manager.Federate.Adjust*, *Manager.Federate.Request*, *Manager.Federate.Report*, and *Manager.Federate.Service*. Specific interactions, sent and received by the RTI, shall be subclasses of these classes and shall be described in the following paragraphs.

Note that the data type of all parameters shall be text; the tables in the following paragraphs that describe interaction parameters present a more specific data type. This more specific type shall represent the data type from which the text data could be generated (itoa in C, for example). For enumerated parameters, the values shall be presented in the form that shall be depicted in the RTI API; specific values shall depend on the computer-language version of the API.

11.2.1 Interaction class *Manager.Federate.Adjust*

The interaction class *Manager.Federate.Adjust* shall permit a federate to adjust the RTI state variables associated with another federate. Interactions that are subclasses of this interaction class shall be

- *SetTiming*
- *ModifyAttributeState*
- *SetServiceReporting*
- *SetExceptionLogging*

11.2.1.1 Interaction subclass *SetTiming*

The interaction subclass *SetTiming* shall adjust the time period between updates of the *Manager.Federate* object instance for the federate. If this interaction is never sent, the RTI shall not perform periodic updates.

Table 6—Interaction subclass *SetTiming*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ReportPeriod	long	Number of seconds between updates of instance attribute values of the <i>Federate</i> object instance (A zero value causes periodic updates to cease.)

11.2.1.2 Interaction subclass *ModifyAttributeState*

The interaction subclass *ModifyAttributeState* shall modify the ownership state of an attribute of an object instance for the federate.

Table 7—Interaction subclass *ModifyAttributeStat*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance whose attribute state is being changed
Attribute	handle	Designator of the instance attribute whose state is being changed
AttributeState	enumerated	Desired state for the attribute of the object instance; valid values are <ul style="list-style-type: none"> — Owned — Unowned

11.2.1.3 Interaction subclass *SetServiceReporting*

The interaction subclass *SetServiceReporting* shall specify whether to report service invocations via *Manager.Federate.Report.ReportServiceInvocation* interactions.

Table 8—Interaction subclass *SetServiceReporting*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ReportingState	boolean	Whether the RTI should report service invocations

11.2.1.4 Interaction subclass *SetExceptionLogging*

The interaction subclass *SetExceptionLogging* shall specify whether to log RTI exceptions to a file.

Table 9—Interaction subclass *SetExceptionLogging*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
LoggingState	boolean	Whether the RTI should log exceptions

11.2.2 Interaction class *Manager.Federate.Request*

The interaction class *Manager.Federate.Request* shall permit a federate to request RTI data about another federate. Interactions that are subclasses of this interaction class shall be

- *RequestPublications*
- *RequestSubscriptions*
- *RequestObjectsOwned*
- *RequestObjectsUpdated*
- *RequestObjectsReflected*
- *RequestUpdatesSent*
- *RequestInteractionsSent*
- *RequestReflectionsReceived*
- *RequestInteractionsReceived*
- *RequestObjectInformation*

11.2.2.1 Interaction subclass *RequestPublications*

The interaction subclass *RequestPublications* shall request that the RTI send report interactions that shall contain the publication data of a federate. It shall result in one interaction of class *Manager.Federate.Report.ReportInteractionPublication* and one interaction of class *Manager.Federate.Report.ReportObjectPublication* for each object class published.

Table 10—Interaction subclass *RequestPublications*

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.2 Interaction subclass *RequestSubscriptions*

The interaction subclass *RequestSubscriptions* shall request that the RTI send report interactions that shall contain the subscription data of a federate. It shall result in one interaction of class *Manager.Federate.-Report.ReportInteractionSubscription* and one interaction of class *Manager.Federate.-Report.ReportObjectSubscription* for each object class published.

Table 11—Interaction subclass *RequestSubscriptions*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.3 Interaction subclass *RequestObjectsOwned*

The interaction subclass *RequestObjectsOwned* shall request that the RTI send a report interaction that shall contain the object ownership data of a federate. It shall result in one interaction of class *Manager.Federate.Report.ReportObjectsOwned*.

Table 12—Interaction subclass *RequestObjectsOwned*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.4 Interaction subclass *RequestObjectsUpdated*

The interaction subclass *RequestObjectsUpdated* shall request that the RTI send a report interaction that shall contain the object updating responsibility of a federate. It shall result in one interaction of class *Manager.Federate.Report.ReportObjectsUpdated*.

Table 13—Interaction subclass *RequestObjectsUpdated*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.5 Interaction subclass *RequestObjectsReflected*

The interaction subclass *RequestObjectsReflected* shall request that the RTI shall send a report interaction that shall contain the objects for which a federate shall reflect updates of instance attributes. It shall result in one interaction of class *Manager.Federate.Report.ReportObjectsReflected*.

Table 14—Interaction subclass *RequestObjectsReflected*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.6 Interaction subclass *RequestUpdatesSent*

The interaction subclass *RequestUpdatesSent* shall request that the RTI send a report interaction that shall contain the number of updates that a federate has generated. It shall result in one interaction of class *Manager.Federate.Report.ReportUpdatesSent* for each transportation type that is used to send updates.

Table 15—Interaction subclass *RequestUpdatesSent*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.7 Interaction subclass *RequestInteractionsSent*

The interaction subclass *RequestInteractionsSent* shall request that the RTI send a report interaction that shall contain the number of interactions that a federate has generated. It shall result in one interaction of class *Manager.Federate.Report.ReportInteractionsSent* for each transportation type that is used to send interactions.

Table 16—Interaction subclass *RequestInteractionsSent*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.8 Interaction subclass *RequestReflectionsReceived*

The interaction subclass *RequestReflectionsReceived* shall request that the RTI send a report interaction that shall contain the number of reflections that a federate has received. It shall result in one interaction of class *Manager.Federate.Report.ReportReflectionsReceived* for each transportation type used in receiving reflections.

Table 17—Interaction subclass *RequestReflectionsReceived*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.9 Interaction subclass *RequestInteractionsReceived*

The interaction subclass *RequestInteractionsReceived* shall request that the RTI send a report interaction that shall contain the number of interactions that a federate has received. It shall result in one interaction of class *Manager.Federate.Report.ReportInteractionsReceived* for each transportation type used in receiving interactions.

Table 18—Interaction subclass *RequestInteractionsReceived*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.2.10 Interaction subclass *RequestObjectInformation*

The interaction subclass *RequestObjectInformation* shall request that the RTI send a report interaction that shall contain the information that a federate shall maintain on a single object instance. It shall result in one interaction of class *Manager.Federate.Report.ReportObjectInformation*.

Table 19—Interaction subclass *RequestObjectInformation*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance for which information is being requested

11.2.3 Interaction class *Manager.Federate.Report*

The interaction class *Manager.Federate.Report* shall report RTI data about a federate. The RTI shall send these interactions in response to interactions of class *Manager.Federate.Request*. Interactions that are subclasses of this interaction class shall be

- *ReportObjectPublication*
- *ReportInteractionPublication*
- *ReportObjectSubscription*
- *ReportInteractionSubscription*
- *ReportObjectsOwned*
- *ReportObjectsUpdated*
- *ReportObjectsReflected*
- *ReportUpdatesSent*
- *ReportReflectionsReceived*
- *ReportInteractionsSent*
- *ReportInteractionsReceived*
- *ReportObjectInformation*
- *Alert*
- *ReportServiceInvocation*

11.2.3.1 Interaction subclass *ReportObjectPublication*

The interaction subclass *ReportObjectPublication* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestPublications*. It shall report the attributes of one object class published by the federate. One of these interactions shall be sent for each object class containing attributes that are published by the federate.

Table 20—Interaction subclass *ReportObjectPublication*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
NumberOfClasses	long	The number of object classes for which the federate publishes attributes
ObjectClass	handle	The object class whose publication is being reported
AttributeList	handle list	Comma-separated list of attributes of ObjectClass that the federate is publishing (null string if none)

11.2.3.2 Interaction subclass *ReportInteractionPublication*

The interaction subclass *ReportInteractionPublication* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestPublications*. It shall report the interaction classes published by the federate.

Table 21—Interaction subclass *ReportInteractionPublication*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClassList	handle list	Comma-separated list of interaction classes that the federate is publishing (null string if none)

11.2.3.3 Interaction subclass *ReportObjectSubscription*

The interaction subclass *ReportObjectSubscription* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestSubscriptions*. It shall report the attributes of one object class subscribed to by the federate. One of these interactions shall be sent for each object class that is subscribed to by the federate.

Table 22—Interaction subclass *ReportObjectSubscription*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
NumberOfClasses	long	The number of object classes for which the federate subscribes to attributes
ObjectClass	handle	The object class whose subscription is being reported
AttributeList	handle/active	Comma-separated list of attribute/subscription type pairs. Each pair consists of a designator of an <i>ObjectClass</i> attribute that the federate is subscribing to and whether the federate is actively subscribing. The attribute is separated from the subscription type by a slash (/) (null string if no subscriptions).

11.2.3.4 Interaction subclass *ReportInteractionSubscription*

The interaction subclass *ReportInteractionSubscription* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestSubscriptions*. It shall report the interaction classes subscribed to by the federate.

Table 23—Interaction subclass *ReportInteractionSubscription*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClassList	handle/active	Comma-separated list of interaction class/subscription type pairs. Each pair consists of the designator of an interaction class that the federate is subscribed to and whether the federate is actively subscribing. The class is separated from the subscription type by a slash (/) (null string if no subscriptions)

11.2.3.5 Interaction subclass *ReportObjectsOwned*

The interaction subclass *ReportObjectsOwned* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestObjectsOwned*. It shall report the number of object instances (by class) whose *PrivilegeToDelete* attribute shall be owned by the federate.

Table 24—Interaction subclass *ReportObjectsOwned*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectCounts	handle/ count, ...	A comma-separated list of object instance counts. Each object instance count consists of an object class designator and the number of object instances of that class. The designator is separated from the number by a slash (/) (null string if no object instances exist).

11.2.3.6 Interaction subclass *ReportObjectsUpdated*

The interaction subclass *ReportObjectsUpdated* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestObjectsUpdated*. It shall report the number of object instances (by class) for which the federate shall be responsible for updating at least one instance attribute; where the federate shall publish the instance attribute, own the attribute of the object instance, and be notified by the RTI that the federate should update the values of the instance attribute.

Table 25—Interaction subclass *ReportObjectsUpdated*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectCounts	handle/ count, ...	Comma-separated list of object instance counts. Each object instance count consists of an object class designator and the number of object instances of that class. The designator is separated from the number by a slash (/) (null string if no object instances).

11.2.3.7 Interaction subclass *ReportObjectsReflected*

The interaction subclass *ReportObjectsReflected* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestObjectsReflected*. It shall report the number of object instances (by class) for which the federate shall reflect updates of at least one attribute.

Table 26—Interaction subclass *ReportObjectsReflected*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectCounts	handle/ count, ...	Comma-separated list of object counts. Each object instance count consists of an object class designator and the number of object instances of that class. The designator is separated from the number by a slash (/) (null string if no object instances).

11.2.3.8 Interaction subclass *ReportUpdatesSent*

The interaction subclass *ReportUpdatesSent* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestUpdatesSent*. It shall report the number of updates sent (by object class) by the federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.

Table 27—Interaction subclass *ReportUpdatesSent*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
TransportationType	enumerated	Transportation type used in sending updates; valid values are — Reliable — Best effort
UpdateCounts	handle/ count, ...	Comma-separated list of update counts. Each update count consists of an object class designator and the number of updates sent of that class. The designator is separated from the number by a slash (/) (null string if no updates).

11.2.3.9 Interaction subclass *ReportReflectionsReceived*

The interaction subclass *ReportReflectionsReceived* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestReflectionsReceived*. It shall report the number of reflections received (by object class) by the federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.

Table 28—Interaction subclass *ReportReflectionsReceived*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
TransportationType	enumerated	Transportation type used in receiving reflections; valid values are — Reliable — Best effort
ReflectCounts	handle/ count, ...	Comma-separated list of reflection counts. Each reflection count consists of an object class designator and the number of reflections received of that class. The designator is separated from the number by a slash (/) (null string if no reflections).

11.2.3.10 Interaction subclass *ReportInteractionsSent*

The interaction subclass *ReportInteractionsSent* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestInteractionsSent*. It shall report the number of interactions sent (by interaction class) by the federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.

Table 29—Interaction subclass *ReportInteractionsSent*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
TransportationType	enumerated	Transportation type used in sending interactions; valid values are — Reliable — Best effort
InteractionCounts	count list	Comma-separated list of interaction counts. Each interaction count consists of an interaction class handle and the number of interactions of that class. The handle is separated from the number by a slash (/) (null string if no interactions).

11.2.3.11 Interaction subclass *ReportInteractionsReceived*

The interaction subclass *ReportInteractionsReceived* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestInteractionsReceived*. It shall report the number of interactions received (by interaction class) by the federate since the beginning of the federation execution. One interaction of this class shall be sent by the RTI for each transportation type used.

Table 30—Interaction subclass *ReportInteractionsReceived*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
TransportationType	enumerated	Transportation type used in receiving interactions; valid values are — Reliable — Best effort
InteractionCounts	count list	Comma-separated list of interaction counts. Each interaction count consists of an interaction class handle and the number of interactions of that class. The handle is separated from the number by a slash (/) (null string if no interactions).

11.2.3.12 Interaction subclass *ReportObjectInformation*

The interaction subclass *ReportObjectInformation* shall be sent by the RTI in response to an interaction of class *Manager.Federate.Request.RequestObjectInformation*. It shall report on a single object instance and portray the attributes of that object instance that shall be owned by the federate, the registered class of the object instance, and the known class of the object instance.

Table 31—Interaction subclass *ReportObjectInformation*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance for which the interaction was sent
OwnedAttributeList	handle list	Comma-separated list of the handles of all instance attributes owned for the object instance by the federate (null string if none).
RegisteredClass	handle	Designator of the registered class of the object instance
KnownClass	handle	Designator of the known class of the object instance (registered if owned registered by the federate, discovered if discovered by the federate)

11.2.3.13 Interaction subclass *Alert*

The interaction subclass *Alert* shall be sent by the RTI when an exception occurs.

Table 32—Interaction subclass *Alert*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

Parameter	Type	Description
AlertSeverity	enumerated	Severity of alert raised by the RTI; valid values are <ul style="list-style-type: none"> — RTI exception — RTI internal error — RTI federate error — RTI warning — RTI diagnostic
AlertDescription	string	Textual description of the alert
AlertID	long	Numerical identifier of the alert

11.2.3.14 Interaction subclass *ReportServiceInvocation*

The interaction subclass *ReportServiceInvocation* shall be sent by the RTI whenever an RTI service is invoked, either by a federate or by the RTI. By default, the RTI shall not send these interactions. Generation may be controlled (turned on or off) by interactions of class *Manager.Federate.Adjust..SetServiceReporting*. The interaction shall always contain the arguments supplied by the service invoker. If the service invocation was successful, the interaction also shall contain the value returned to the invoker (if the service returns a value); otherwise, the interaction also shall contain an indication of the exception that shall be raised to the invoker.

Table 33—Interaction subclass *ReportServiceInvocation*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Service	string	Textual name of the service
Initiator	enumerated	Initiator of the RTI service; valid values are <ul style="list-style-type: none"> — Federate — RTI
SuccessIndicator	boolean	Whether the service invocation was successful. Exception values are returned along with a false value.
SuppliedArgument1	string	Textual depiction of the first argument supplied in the service invocation
SuppliedArgument2	string	Textual depiction of the second argument supplied in the service invocation
SuppliedArgument3	string	Textual depiction of the third argument supplied in the service invocation
SuppliedArgument4	string	Textual depiction of the fourth argument supplied in the service invocation
SuppliedArgument5	string	Textual depiction of the fifth argument supplied in the service invocation

Parameter	Type	Description
ReturnedArgument	string	Textual depiction of the argument returned by the service invocation (null if the service does not normally return a value or if <i>SuccessIndicator</i> is false)
ExceptDescription	string	Textual description of the exception raised by this service invocation. (null if <i>SuccessIndicator</i> is true.)
ExceptID	long	Numerical identifier of the exception raised by this service invocation (null if <i>SuccessIndicator</i> is true)

11.2.4 Interaction class *Manager.Federate.Service*

The interaction class *Manager.Federate.Service* shall be acted upon by the RTI. These services shall invoke RTI services on behalf of another federate. For services that shall be normally invoked by a federate, they shall cause the RTI to react as if the service has invoked the federate. For services that are normally callbacks from the RTI to a federate, they shall cause the RTI to invoke the callback.

If exceptions arise as a result of the use of these interactions, they shall be reported via the *Manager.Federate.Report.Alert* interaction to all federates that subscribe to this interaction.

NOTE—These interactions shall have the potential to disrupt normal federation execution and should be used with great care.

Interactions that shall be subclasses of this interaction class are

- *ResignFederationExecution*
- *AnnounceSynchronizationPoint*
- *SynchronizationPointAchieved*
- *InitiateFederateSave*
- *FederateSaveBegun*
- *FederateSaveComplete*
- *InitiateFederateRestore*
- *FederateRestoreComplete*
- *PublishObjectClass*
- *UnpublishObjectClass*
- *PublishInteractionClass*
- *UnpublishInteractionClass*
- *SubscribeObjectClassAttributes*
- *UnsubscribeObjectClass*
- *SubscribeInteractionClass*
- *UnsubscribeInteractionClass*
- *DeleteObjectInstance*
- *LocalDeleteObjectInstance*
- *ChangeAttributeTransportationType*
- *ChangeAttributeOrderType*
- *ChangeInteractionTransportationType*
- *ChangeInteractionOrderType*
- *UnconditionalAttributeOwnershipDivestiture*
- *EnableTimeRegulation*
- *DisableTimeRegulation*
- *EnableTimeConstrained*

- *DisableTimeConstrained*
- *EnableAsynchronousDelivery*
- *DisableAsynchronousDelivery*
- *ModifyLookahead*
- *TimeAdvanceRequest*
- *TimeAdvanceRequestAvailable*
- *NextEventRequest*
- *NextEventRequestAvailable*
- *FlushQueueRequest*

11.2.4.1 Interaction subclass *ResignFederationExecution*

The interaction subclass *ResignFederationExecution* shall cause the federate to resign from the federation execution.

Table 34—Interaction subclass *ResignFederationExecution*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ResignAction	enumerated	Action that the RTI is to take in conjunction with the resignation; valid values are <ul style="list-style-type: none"> — Release ownership of all owned instance attributes — Delete all object instances for which the federate has the delete privilege — Perform the first action above, then the second — Perform no actions

11.2.4.2 Interaction subclass *AnnounceSynchronizationPoint*

The interaction subclass *AnnounceSynchronizationPoint* shall cause the federate take action based on the supplied tag.

Table 35—Interaction subclass *InitiatePause*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Label	string	Unique label associated with the synchronization point
Tag	string	Tag specifying the action to be performed by the federate at the synchronizaton point

11.2.4.3 Interaction subclass *SynchronizationPointAchieved*

The interaction subclass *SynchronizationPointAchieved* shall mimic the federate's report of achieving a synchronization point.

Table 36—Interaction subclass *PauseAchieved*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Label	string	Label associated with the synchronization point

11.2.4.4 Interaction subclass *InitiateFederateSave*

The interaction subclass *InitiateFederateSave* shall cause the federate to initiate a save.

Table 37—Interaction subclass *InitiateFederateSave*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Label	string	Label associated with the save

11.2.4.5 Interaction subclass *FederateSaveBegun*

The interaction subclass *FederateSaveBegun* shall mimic the federate's report of starting a save.

Table 38—Interaction subclass *FederateSaveBegun*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.6 Interaction subclass *FederateSaveComplete*

The interaction subclass *FederateSaveComplete* shall mimic the federate's report of completion of a save.

Table 39—Interaction subclass *FederateSaveComplete*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
SuccessIndicator	boolean	Whether the save was successful

11.2.4.7 Interaction subclass *InitiateFederateRestore*

The interaction subclass *InitiateFederateRestore* shall cause the federate to initiate a restore.

Table 40—Interaction subclass *InitiateFederateRestore*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Label	string	Label associated with the restore
NewFederate	string	Federate designator to be used for the restore

11.2.4.8 Interaction subclass *FederateRestoreComplete*

The interaction subclass *FederateRestoreComplete* shall mimic the federate's report of completion of a restore.

Table 41—Interaction subclass *FederateRestoreComplete*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Label	string	Label associated with the restore
SuccessIndicator	boolean	Whether the restore was successful

11.2.4.9 Interaction subclass *PublishObjectClass*

The interaction subclass *PublishObjectClass* shall set the federate's publication status of attributes belonging to an object class.

Table 42—Interaction subclass *PublishObjectClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectClass	handle	Object class for which the federate's publication is set
AttributeList	handle list	Comma-separated list of handles of attributes of <i>ObjectClass</i> , which the federate shall now publish (null string if none) NOTE—A null string implies that the federate shall now publish no attributes.

11.2.4.10 Interaction subclass *UnpublishObjectClass*

The interaction subclass *UnpublishObjectClass* shall cause the federate no longer to publish attributes of an object class.

Table 43—Interaction subclass *UnpublishObjectClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectClass	handle	Object class that the federate shall no longer publish

11.2.4.11 Interaction subclass *PublishInteractionClass*

The interaction subclass *PublishInteractionClass* shall set the federate's publication status of an interaction class.

Table 44—Interaction subclass *PublishInteractionClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClass	handle	Interaction class that the federate shall publish

11.2.4.12 Interaction subclass *UnpublishInteractionClass*

The interaction subclass *UnpublishInteractionClass* shall cause the federate no longer to publish an interaction class.

Table 45—Interaction subclass *UnpublishInteractionClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

Parameter	Type	Description
InteractionClass	handle	Interaction class that the federate shall no longer publish

11.2.4.13 Interaction subclass *SubscribeObjectClassAttributes*

The interaction subclass *SubscribeObjectClassAttributes* shall set the federate's subscription status of attributes belonging to an object class.

Table 46—Interaction subclass *SubscribeObjectClassAttributes*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectClass	handle	Object class for which the federate's subscription shall change
AttributeList	handle list	Comma-separated list of handles of attributes of <i>ObjectClass</i> to which the federate shall now subscribe (null string if none) NOTE—A null string implies that the federate shall now subscribe to no attributes.
Active	boolean	Whether the subscription is active

11.2.4.14 Interaction subclass *UnsubscribeObjectClass*

The interaction subclass *UnsubscribeObjectClass* shall cause the federate no longer to subscribe to attributes of an object class.

Table 47—Interaction subclass *UnsubscribeObjectClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectClass	handle	Object class to which the federate shall no longer be subscribed

11.2.4.15 Interaction subclass *SubscribeInteractionClass*

The interaction subclass *SubscribeInteractionClass* shall set the federate's subscription status to an interaction class.

Table 48—Interaction subclass *SubscribeInteractionClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClass	handle	Interaction class to which the federate shall subscribe

Parameter	Type	Description
Active	boolean	Whether the subscription is active

11.2.4.16 Interaction subclass *UnsubscribeInteractionClass*

The interaction subclass *UnsubscribeInteractionClass* shall cause the federate no longer to subscribe to an interaction class.

Table 49—Interaction subclass *UnsubscribeInteractionClass*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClass	handle	Interaction class to which the federate will no longer subscribe

11.2.4.17 Interaction subclass *DeleteObjectInstance*

The interaction subclass *DeleteObjectInstance* shall cause an object instance to be deleted from the federation.

Table 50—Interaction subclass *DeleteObjectInstance*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance that is to be deleted
Tag	string	Tag associated with the deletion
FederationTime	time	Federation time of the deletion (optional)

11.2.4.18 Interaction subclass *LocalDeleteObjectInstance*

The interaction subclass *LocalDeleteObjectInstance* shall inform the RTI that it shall treat the specified object instance as if the RTI had never notified the affected federate to discover the object instance.

Table 51—Interaction subclass *LocalDeleteObjectInstance*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance that is to be deleted

11.2.4.19 Interaction subclass *ChangeAttributeTransportationType*

The interaction subclass *ChangeAttributeTransportationType* shall change the transportation type used by the federate when sending attributes belonging to a single object instance.

Table 52—Interaction subclass *ChangeAttributeTransportationType*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance whose attribute transportation type is to be changed
AttributeList	handle list	Comma-separated list of the handles of instance attributes whose transportation type is to be changed (null string if none)
TransportationType	enumerated	Transportation type desired for use in updating instance attributes in the list; valid values are — Reliable — Best effort

11.2.4.20 Interaction subclass *ChangeAttributeOrderType*

The interaction subclass *ChangeAttributeOrderType* shall change the ordering type used by the federate when sending attributes belonging to a single object instance.

Table 53—Interaction subclass *ChangeAttributeOrderType*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance whose attribute ordering type is to be changed
AttributeList	handle list	Comma-separated list of the handles of instance attributes whose ordering type is to be changed (null string if none)
OrderingType	enumerated	Ordering type desired for use in sending the instance attribute list; valid values are — Receive — Timestamp

11.2.4.21 Interaction subclass *ChangeInteractionTransportationType*

The interaction subclass *ChangeInteractionTransportationType* shall change the transportation type used by the federate when sending a class of interaction.

Table 54—Interaction subclass *ChangeInteractionTransportationType*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClass	handle	Interaction class whose transportation type is changed by this service invocation

Parameter	Type	Description
TransportationType	enumerated	Transportation type desired for use in sending the interaction class; valid values are — Reliable — Best effort

11.2.4.22 Interaction subclass *ChangeInteractionOrderType*

The interaction subclass *ChangeInteractionOrderType* shall change the ordering type used by the federate when sending a class of interaction.

Table 55—Interaction subclass *ChangeInteractionOrderType*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
InteractionClass	handle	Interaction class whose ordering type is changed by this service invocation
OrderingType	enumerated	Ordering type desired for use in sending the interaction class; valid values are — Receive — Timestamp

11.2.4.23 Interaction subclass *UnconditionalAttributeOwnershipDivestiture*

The interaction subclass *UnconditionalAttributeOwnershipDivestiture* shall cause the ownership of attributes contained in an object instance to be unconditionally divested by the federate.

Table 56—Interaction subclass *UnconditionalAttributeOwnershipDivestiture*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
ObjectInstance	string	Name of the object instance whose attributes' ownership is to be divested
AttributeList	handle list	Comma-separated list of handles of instance attributes belonging to <i>ObjectInstance</i> whose ownership is to be divested by the federate (null string if none)

11.2.4.24 Interaction subclass *EnableTimeRegulation*

The interaction subclass *EnableTimeRegulation* shall cause the federate to begin regulating the logical time of other federates.

Table 57—Interaction subclass *EnableTimeRegulation*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time at which time regulation is to begin
Lookahead	time	Lookahead to be used by the federate while regulating other federates

11.2.4.25 Interaction subclass *DisableTimeRegulation*

The interaction subclass *DisableTimeRegulation* shall cause the federate to cease regulating the logical time of other federates.

Table 58—Interaction subclass *DisableTimeRegulation*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.26 Interaction subclass *EnableTimeConstrained*

The interaction subclass *EnableTimeConstrained* shall cause the logical time of the federate to begin being constrained by the logical times of other federates.

Table 59—Interaction subclass *EnableTimeConstrained*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.27 Interaction subclass *DisableTimeConstrained*

The interaction subclass *DisableTimeConstrained* shall cause the logical time of the federate to cease being constrained by the logical times of other federates.

Table 60—Interaction subclass *DisableTimeConstrained*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.28 Interaction subclass *EnableAsynchronousDelivery*

The interaction subclass *EnableAsynchronousDelivery* shall cause the RTI to deliver receive-order messages to the federate when its time manager state is either “Time Pending” or “Idle.” The federate shall be time-constrained for this interaction to have effect.

Table 61—Interaction subclass *EnableAsynchronousDelivery*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.29 Interaction subclass *DisableAsynchronousDelivery*

The interaction subclass *DisableAsynchronousDelivery* shall cause the RTI to deliver receive-order messages to the federate only when its time manager state is “Time Pending.” The federate shall be time-constrained for this interaction to have effect.

Table 62—Interaction subclass *EnableAsynchronousDelivery*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining

11.2.4.30 Interaction subclass *ModifyLookahead*

The interaction subclass *ModifyLookahead* shall change the lookahead value used by the federate.

Table 63—Interaction subclass *ModifyLookahead*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
Lookahead	time	New value for lookahead

11.2.4.31 Interaction subclass *TimeAdvanceRequest*

The interaction subclass *TimeAdvanceRequest* shall request an advance of the federate's logical time on behalf of the federate, and release zero or more messages for delivery to the federate.

Table 64—Interaction subclass *TimeAdvanceRequest*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time requested

11.2.4.32 Interaction subclass *TimeAdvanceRequestAvailable*

The interaction subclass *TimeAdvanceRequestAvailable* shall request an advance of the federate's logical time, on behalf of the federate, and release zero or more messages for delivery to the federate.

Table 65—Interaction subclass *TimeAdvanceRequestAvailable*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time requested

11.2.4.33 Interaction subclass *NextEventRequest*

The interaction subclass *NextEventRequest* shall request the logical time of the federate to be advanced to the time stamp of the next TSO message that shall be delivered to the federate, provided that the message shall have a time stamp no greater than the logical time specified in the request.

Table 66—Interaction subclass *NextEventRequest*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time requested

11.2.4.34 Interaction subclass *NextEventRequestAvailable*

The interaction subclass *NextEventRequestAvailable* shall request the logical time of the federate to be advanced to the time stamp of the next TSO message that shall be delivered to the federate, provided that the message shall have a time stamp no greater than the logical time specified in the request.

Table 67—Interaction subclass *NextEventRequestAvailable*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time requested

11.2.4.35 Interaction subclass *FlushQueueRequest*

The interaction subclass *FlushQueueRequest* shall request the logical time of the federate to be advanced to the time stamp of the next TSO message that shall be delivered to the federate, provided that the message shall have a time stamp no greater than the logical time specified in the request. All TSO messages shall be delivered to the federate.

Table 68—Interaction subclass *FlushQueueRequest*

Parameter	Type	Description
Federate	handle	Designator of the affected federate that was provided when joining
FederationTime	time	Federation time requested

12. Federation execution data (FED)

12.1 FED data interchange format (FED DIF)

The high-level architecture FED data interchange format (DIF) is a standard file-exchange format that shall be used to store and transfer HLA FED files between multiple tools including object-model development tools (OMDTs) and RTIs.

12.1.1 BNF notation of the DIF

To ensure that there is no ambiguity in the definition of the DIF, the DIF shall be defined terms of Backus-Naur Form (BNF). BNF is a formal notation used to describe inductive specifications. Attributed to John Backus and Peter Naur, it was invented to describe the syntax of Algol 60 in an unambiguous manner. Since then it has become widely accepted and used by most authors of books on new programming languages to specify the syntax rules of the language.

Because no standard BNF notation exists, it is necessary to present the conventions for the notation used here. This document will use extended BNF (EBNF), which includes some additional constructs to handle iteration and alternation as described in the following sub-clauses.

12.1.1.1 BNF notation conventions

BNF has three major parts:

- Terminals, which shall require no further definition,
- Non-terminals, which shall be defined in terms of other non-terminals and terminals, and
- Productions which, for each non-terminal, shall precisely state how the non-terminal is constructed.

Certain symbols within the BNF shall have special meanings. These shall be called *meta-symbols*, and they shall be used to structure the BNF. Double quotes, angle brackets, braces, etc., shall be meta-symbols within BNF, and their definition and use will be given below.

- Words inside double quotes (“word”) shall represent literal words themselves (these shall be called *terminals*).
- Words contained within angle brackets ‘< >’ shall represent semantic categories (i.e. *non-terminals*) that shall be resolved by reading their definition elsewhere in the BNF. An example of a non-terminal is <NameCharacter>.
- A production (sometimes called a rule) shall be a statement of the definition of a non-terminal. It shall be designated by the production meta-symbol ‘::=’, which shall assign the definition to the right-hand side (RHS) of the production to the non-terminal on the left hand side (LHS) of the production symbol. The LHS shall always consist of a single non-terminal, while the RHS may consist of any combination of terminals and non-terminals. The symbol ‘::=’ shall be read as “...is defined to be...” or “...is composed of...”. An example of a production is:

```
<<SpaceName>> ::= <NameString>;
```

- Selection of one item for a given instance shall be designated by use of the vertical bar symbol ‘|’. The symbol ‘|’ shall be read as “...or...”.

- Each BNF statement shall be terminated by a semicolon (;).

12.1.1.2 EBNF notation conventions

- Terminals shall be represented using words inside double quotes. In addition, terminals shall be further highlighted using **boldfaced** text. An example of a terminal is "**Federation**".
- The BNF used in this standard shall add a special case of non-terminal that shall be denoted by double brackets '<< >>' rather than single angle brackets. This special case non-terminal shall be a reference to the FED DIF glossary found in clause 12.1.4.
- Optional Items shall be enclosed by square bracket meta-symbols '[' and ']'. Square brackets shall indicate that the item exists either zero or one time; that is, it may or may not exist. An example of an optional item is [**<SpaceName>**], which indicates that the SpaceName item may or may not be present in the DIF.
- Repetition (zero, one, or many) shall be performed by the curly brace meta-symbols '{' and '}'.
 - Curly braces followed by a * character shall indicate that there are zero or more sequential instances of the item.
 - Curly braces followed by a + character shall indicate that there shall be one or more sequential instances of the item.
- The double period .. used within a literal shall be a shortcut notation for denoting the set of ASCII characters between the characters to either side of them. An example of this is "**a..z**", which denotes the set of lowercase letters between 'a' and 'z' inclusive.

12.1.1.3 Basic BNF constructs

The following are a set of basic BNF constructs referenced in the main body of the DIF BNF definition. They are defined separately to make the main body more readable.

```
<NameString> ::= <Letter> {<NameCharacter>}*;

<NameCharacter> ::= <Letter> | <DecimalDigit> | "_" | "+" | "-" | "*" | "/" |

    "@" | "$" | "%" | "^" | "&" | "=" | "<" | ">" | "~" | "!" | "#";

<Letter> ::= "a..z" | "A..Z";

<DecimalDigit> ::= "0..9";
```

Figure 16—Basic BNF constructs

12.1.2 HLA FED DIF BNF definition

```

<HLA-FED-DIF-v1.3> ::= "(FED " <Federation> <FEDversion> <Spaces> <ObjectClasses>
                        <InteractionClasses> ")";

<Federation> ::= "(Federation " <<FEDname>> ")";

<FEDversion> ::= "(FEDversion " <<FEDDIFversionNumber>> ")";

<<FEDname>> ::= <NameString>;

<<FEDDIFversionNumber>> ::= "v1.3";

<Spaces> ::= "(spaces " {<Space>}* ")";

<Space> ::= "(space " <<SpaceName>> {<Dimension>}* ")";

<Dimension> ::= "(dimension " <<DimensionName>> ")";

<<SpaceName>> ::= <NameString>;

<<DimensionName>> ::= <NameString>;

<ObjectClasses> ::= "(objects "
                    "(class ObjectRoot "
                    "(attribute    privilegeToDelete    " <<Transport>>    <<Order>>
[<<SpaceName>>] ") "
                    "(class RTIprivate)"
                    {<ObjectClass>}* ")");

<ObjectClass> ::= "(class " <<ObjectClassName>>
                  {<Attribute>}*
                  {<ObjectClass>}* ")";

<Attribute> ::= "(attribute " <<AttributeName>> <<Transport>> <<Order>> [<<SpaceName>>]
                ")";

<<ObjectClassName>> ::= <NameString>;

<<Transport>> ::= <NameString>;

```



```

<<Order>> ::= <NameString>;

<<AttributeName>> ::= <NameString>;

<InteractionClasses> ::= "(interactions "
    "(class InteractionRoot " <<Transport>> <<Order>> [<<SpaceName>>]
    "(class RTIprivate " <<Transport>> <<Order>> [<<SpaceName>>] ")"
    {InteractionClass}* ")");

<InteractionClass> ::= "(class " <<InteractionClassName>> <<Transport>> <<Order>>
[<<SpaceName>>]
    {<Parameter>}*
    {<InteractionClass>}* ")" ;

<Parameter> ::= "(parameter " <<ParameterName>> ")" ;

<<InteractionClassName>> ::= <NameString>;

<<ParameterName>> ::= <NameString>;

```

Figure 17—HLA FED DIF BNF definition

12.1.3 FED DIF meta-data consistency

The use of BNF cannot completely capture all of the rules that specify a complete and correct DIF file or object model. A FED DIF file shall comply with the following rules to be complete, consistent, and correct:

1. A comment shall be prefixed with two semicolons and terminated by \n (; *comment* \n).
2. A comment may appear at the beginning of a line (on a line by itself).
3. A comment may appear at the end of a line following a FED element.
4. Wherever a literal space appears in the DIF definition, multiple spaces shall be valid.
5. One or more literal spaces shall be allowed between any parenthesis and the adjoining text.
6. Use of routing spaces shall optional.
7. Routing space names within a FED file shall be unique.
8. Dimension names within a single routing space shall be unique.
9. All names shall be case insensitive.
10. Object- and interaction-class names shall be unique where they share a common parent class. Class names may be reused across multiple branches or tiers of the class hierarchy, as long as no two sibling classes have the same name.
11. All MOM object and interaction classes along with their attributes and parameters shall be included in each FED DIF file.
12. All terminals in the BNF description and DIF files produced in accordance with this BNF description shall be considered to be case insensitive. For example, the literal "ObjectModel" and "OBJECTMODEL" shall be considered equivalent. Capitalization shall be used in the BNF strictly to enhance readability.

12.1.4 FED DIF glossary

This glossary shall define the terms used in the HLA FED DIF BNF definition to the corresponding concepts in the main body of the interface specification.

AttributeName	The name of an object-class attribute.
DimensionName	The name of a routing-space dimension.
FEDDIFversionNumber	The identifier for a specific version of the FED DIF.
FEDname	The name of an HLA federation.
InteractionClassName	The name of an interaction class.
ObjectClassName	The name of an object class.
Order	The name of a type of message ordering.
ParameterName	The name of an interaction-class parameter.
SpaceName	The name of a routing space.
Transport	The name of a type of message transportation.

12.2 Example FED file

Figure 18 depicts a complete FED file with particular emphasis on the MOM (MOM definitions are complete). Several liberties have been taken with the depiction:

- Aspects of the file that should be completed for a specific federation execution are in *italics*. This includes definition of space characteristics, specification of transportation and order type, and optionally space characteristic for each class attribute and interaction class. It also includes definition of extensions to the MOM object and interaction classes and specification of federation object and interaction classes.
- The **x** characters have been added to aid the user in associating subclasses with classes and attributes with classes.

Figure 18—FED file with MOM definitions

```

(FED
  (Federation MOM)
  (FEDversion v1.3)
(spaces
  Space definitions
)
(objects
  x (class objectRoot
  x x (attribute privilegeToDelete transport order space)
  x x (class RTIprivate)
  x x (class Manager
  x x x (class Federate
  x x x (attribute Federate transport order space)
  x x x (attribute FederateType transport order space)
  x x x (attribute FederateName transport order space)
  x x x (attribute FederateHost transport order space)
  x x x (attribute RTIversion transport order space)
  x x x (attribute FEDid transport order space)
  x x x (attribute TimeConstrained transport order space)
  x x x (attribute TimeRegulating transport order space)
  x x x (attribute AsynchronousDelivery transport order space)
  x x x (attribute FederateState transport order space)
  x x x (attribute TimeManagerState transport order space)
  x x x (attribute FederateTime transport order space)
  x x x (attribute Lookahead transport order space)

```

```

x x x (attribute LBTS transport order space)

x x x (attribute MinNextEventTime transport order space)

x x x (attribute ROLength transport order space)

x x x (attribute TSOlength transport order space)

x x x (attribute ReflectionsReceived transport order space)

x x x (attribute UpdatesSent transport order space)

x x x (attribute InteractionsReceived transport order space)

x x x (attribute InteractionsSent transport order space)

x x x (attribute ObjectsOwned transport order space)

x x x (attribute ObjectsUpdated transport order space)

x x x (attribute ObjectsReflected transport order space) )

x x x (class Federation

x x x (attribute FederationName transport order space)

x x x (attribute FederatesInFederation transport order space)

x x x (attribute RTIversion transport order space)

    x x x (attribute LastSaveName transport order space)

x x x (attribute LastSaveTime transport order space)

x x x (attribute NextSaveName transport order space)

x x x (attribute NextSaveTime transport order space) )

x x x ( MOM Object Class extension definitions )

x x )

x x ( User Object Class definitions )

x )

)

(interactions

x (class interactionRoot transport order space

x x (class RTIprivate transport order space)

```

```

x x (class Manager transport order space

x x x (class Federate transport order space

x x x x (parameter Federate)

x x x x (class Request transport order space

x x x x x (class RequestPublications transport order space )

x x x x x (class RequestSubscriptions transport order space )

x x x x x (class RequestObjectsOwned transport order space )

x x x x x (class RequestObjectsUpdated transport order space )

x x x x x (class RequestObjectsReflected transport order space )

x x x x x (class RequestUpdatesSent transport order space )

x x x x x (class RequestInteractionsSent transport order space )

x x x x x (class RequestReflectionsReceived transport order space )

x x x x x (class RequestInteractionsReceived transport order space )

x x x x x (class RequestObjectInformation transport order space

x x x x x (parameter ObjectInstance) )

x x x x )

x x x x (class Report transport order space

x x x x x (class ReportObjectPublication transport order space

x x x x x (parameter NumberOfClasses)

x x x x x (parameter ObjectClass)

x x x x x (parameter AttributeList) )

x x x x x (class ReportInteractionPublication transport order space

x x x x x (parameter InteractionClassList) )

x x x x x (class ReportObjectSubscription transport order space

x x x x x (parameter NumberOfClasses)

x x x x x (parameter ObjectClass)

x x x x x (parameter AttributeList) )

```

```

x x x x x (class ReportInteractionSubscription transport order space
x x x x x (parameter InteractionClassList) )

x x x x x (class ReportObjectsOwned transport order space
x x x x x (parameter ObjectCounts) )

x x x x x (class ReportObjectsUpdated transport order space
x x x x x (parameter ObjectCounts) )

x x x x x (class ReportObjectsReflected transport order space
x x x x x (parameter ObjectCounts) )

x x x x x (class ReportUpdatesSent transport order space
x x x x x (parameter TransportType)
x x x x x (parameter UpdateCounts) )

x x x x x (class ReportReflectionsReceived transport order space
x x x x x (parameter TransportType)
x x x x x (parameter ReflectCounts) )

x x x x x (class ReportInteractionsSent transport order space
x x x x x (parameter TransportType)
x x x x x (parameter InteractionCounts) )

x x x x x (class ReportInteractionsReceived transport order space
x x x x x (parameter TransportType)
x x x x x (parameter InteractionCounts) )

x x x x x (class ReportObjectInformation transport order space
x x x x x (parameter ObjectInstance)
x x x x x (parameter OwnedAttributeList)
x x x x x (parameter RegisteredClass)
x x x x x (parameter KnownClass) )

x x x x x (class Alert transport order space
x x x x x (parameter AlertSeverity)
x x x x x (parameter AlertDescription)

```

```

x x x x x (parameter AlertID) )

x x x x x (class ReportServiceInvocation transport order space

x x x x x (parameter Service)

x x x x x (parameter Initiator)

x x x x x (parameter SuccessIndicator)

x x x x x (parameter SuppliedArgument1)

x x x x x (parameter SuppliedArgument2)

x x x x x (parameter SuppliedArgument3)

x x x x x (parameter SuppliedArgument4)

x x x x x (parameter SuppliedArgument5)

x x x x x (parameter ReturnedArgument)

    x x x x x (parameter ExceptDescription)

    x x x x x (parameter ExceptID) )

x x x x )

x x x x (class Adjust transport order space

x x x x x (class SetTiming transport order space

x x x x x (parameter ReportPeriod) )

x x x x x (class ModifyAttributeState transport order space

x x x x x (parameter ObjectInstance)

x x x x x (parameter Attribute)

x x x x x (parameter AttributeState) )

x x x x x (class SetServiceReporting transport order space

x x x x x (parameter ReportingState) )

x x x x x (class SetExceptionLogging transport order space

x x x x x (parameter LoggingState) )

x x x x )

x x x x (class Service transport order space

x x x x x (class ResignFederationExecution transport order space

```

```

x x x x x (parameter ResignKey) )

x x x x x (class AnnounceSynchronizationPoint transport order space

x x x x x (parameter Label)

    x x x x x (parameter Tag) )

x x x x x (class SynchronizationPointAchieved transport order space

x x x x x (parameter Label) )

x x x x x (class InitiateFederateSave transport order space

x x x x x (parameter Label) )

x x x x x (class FederateSaveBegun transport order space )

x x x x x (class FederateSaveComplete transport order space

x x x x x (parameter SuccessIndicator) )

x x x x x (class InitiateFederateRestore transport order space

x x x x x (parameter Label) )

x x x x x (class FederateRestoreComplete transport order space

x x x x x (parameter SuccessIndicator) )

x x x x x (class PublishObjectClass transport order space

x x x x x (parameter ObjectClass)

x x x x x (parameter AttributeList) )

x x x x x (class UnpublishObjectClass transport order space

x x x x x (parameter ObjectClass) )

x x x x x (class PublishInteractionClass transport order space

x x x x x (parameter InteractionClass) )

x x x x x (class UnpublishInteractionClass transport order space

x x x x x (parameter InteractionClass) )

x x x x x (class SubscribeObjectClassAttributes transport order space

x x x x x (parameter ObjectClass)

x x x x x (parameter AttributeList)

    x x x x x (parameter Active) )

```



```

x x x x x (class UnsubscribeObjectClass transport order space
x x x x x (parameter ObjectClass) )

x x x x x (class SubscribeInteractionClass transport order space
x x x x x (parameter InteractionClass)
      x x x x x (parameter Active) )

x x x x x (class UnsubscribeInteractionClass transport order space
x x x x x (parameter InteractionClass) )

x x x x x (class DeleteObjectInstance transport order space
x x x x x (parameter ObjectInstance)
x x x x x (parameter Tag)
      x x x x x (parameter FederationTime) )

x x x x x (class LocalDeleteObjectInstance transport order space
x x x x x (parameter ObjectInstance) )

x x x x x (class ChangeAttributeTransportationType transport order space
x x x x x (parameter ObjectInstance)
x x x x x (parameter AttributeList)
x x x x x (parameter TransportationType) )

x x x x x (class ChangeAttributeOrderType transport order space
x x x x x (parameter ObjectInstance)
x x x x x (parameter AttributeList)
x x x x x (parameter OrderingType) )

x x x x x (class ChangeInteractionTransportationType transport order space
x x x x x (parameter InteractionClass)
x x x x x (parameter TransportationType) )

x x x x x (class ChangeInteractionOrderType transport order space
x x x x x (parameter InteractionClass)
x x x x x (parameter OrderingType) )

x x x x x (class UnconditionalAttributeOwnershipDivestiture transport order space

```

```

x x x x x (parameter ObjectInstance)

x x x x x (parameter AttributeList) )

x x x x x (class EnableTimeRegulation transport order space
x x x x x (parameter FederationTime)
x x x x x (parameter Lookahead) )

x x x x x (class DisableTimeRegulation transport order space )
x x x x x (class EnableTimeConstrained transport order space )
x x x x x (class DisableTimeConstrained transport order space )

x x x x x (class EnableAsynchronousDelivery transport order space )
x x x x x (class DisableAsynchronousDelivery transport order space )

x x x x x (class ModifyLookahead transport order space
x x x x x (parameter Lookahead) )

x x x x x (class TimeAdvanceRequest transport order space
x x x x x (parameter FederationTime) )

x x x x x (class TimeAdvanceRequestAvailable transport order space
x x x x x (parameter FederationTime) )

x x x x x (class NextEventRequest transport order space
x x x x x (parameter FederationTime) )

x x x x x (class NextEventRequestAvailable transport order space
x x x x x (parameter FederationTime) )

x x x x x (class FlushQueueRequest transport order space
x x x x x (parameter FederationTime) )

x x x x x )

x x x x )

x x x ( MOM Interaction Class extension definitions )

x x x )

x x x ( User Interaction Class definitions )

```

<pre>x)))</pre>
--

ANNEX A (normative)

IDL application programmer's interface

TBS

ANNEX B (normative)

C++ application programmer's interface

```
//File RTI.hh

#ifndef RTI_hh
#define RTI_hh

#include <fstream.h>
#include <math.h>
#include <rw/rwfile.h>

struct RTIambPrivateRefs;
struct RTIambPrivateData;

class RTI {
public:

#include "baseTypes.hh"
#include "RTItypes.hh"

    class RTIambassador {
    public:
#include "RTIambServices.hh"
        RTIambPrivateData* privateData;
    private:
        RTIambPrivateRefs* privateRefs;
    };

    class FederateAmbassador {
    public:
#include "federateAmbServices.hh"
    };
};

#endif
```

```

//File baseTypes.hh
//Included in RTI.hh

#ifndef NULL
#define NULL (0)
#endif

typedef unsigned short UShort;
typedef short Short;
typedef unsigned long ULong;
typedef long Long;
typedef double Double;
typedef float Float;

enum Boolean {
    RTI_FALSE = 0,
    RTI_TRUE};

class Exception {
public:
    RTI::ULong _serial;
    char *_reason;
    const char *_name;
    Exception (const char *reason);
    Exception (RTI::ULong serial, const char *reason=NULL);
    Exception (const Exception &toCopy);
    virtual ~Exception ();
    Exception & operator = (const Exception &);
    friend ostream& operator<< (ostream &, Exception *);
};

#define RTI_EXCEPT(A) \
class A : public RTI::Exception { \
public: \
    static const char *_ex; \
    A (const char *reason) : Exception (reason) { _name = _ex; } \
    A (RTI::ULong serial, const char *reason=NULL) \
    : Exception (serial, reason) { _name = _ex; } \
    A (const RTI::Exception &toCopy) : Exception(toCopy) { _name = _ex; } \
};

```

```

// File RTItypesypes.hh
// Included in RTI.hh

//
//          RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by pointer value.
// C3  Function return by value.
// C4  In parameter by const pointer value.  Caller provides memory.
//      Caller may free memory or overwrite it upon completion of
//      the call.  Callee must copy during the call anything it
//      wishes to save beyond completion of the call.  Parameter
//      type must define const accessor methods.
// C5  Out parameter by pointer value.  Caller provides reference to object.
//      Callee constructs an instance on the heap (new) and returns.
//      The caller destroys the instance (delete) at its leisure.
// C6  Function return by pointer value.  Callee constructs an instance on
//      the heap (new) and returns a reference.  The caller destroys the
//      instance (delete) at its leisure.

#define MAX_FEDERATION          32
#define MAX_FEDERATE            32
#define MAX_NAME_LENGTH        64
#define MAX_SPACES              10
#define MAX_OBJECT_CLASSES      200
#define MAX_INTERACTION_CLASSES 200
#define MAX_ATTRIBUTES_PER_CLASS 200
#define MAX_PARAMETERS_PER_CLASS 200
#define MAX_DIMENSIONS_PER_SPACE 10
#define DEFAULT_SPACE_NAME      "defaultSpace"
#define MAX_USER_TAG_LENGTH     30
#define RTI_VERSION              "1.1R3"
#define MAX_EXTENT               ((RTI::ULong) 0xc0000000)
#define MIN_EXTENT               ((RTI::ULong) 0x40000000)
#define POSITIVE_INFINITY        (HUGE_VAL)    //see <math.h>

RTI_EXCEPT(ArrayIndexOutOfBounds)
RTI_EXCEPT(AttributeAcquisitionWasNotRequested)
RTI_EXCEPT(AttributeAcquisitionWasNotCanceled)
RTI_EXCEPT(AttributeAlreadyBeingAcquired)
RTI_EXCEPT(AttributeAlreadyBeingDivested)
RTI_EXCEPT(AttributeAlreadyOwned)
RTI_EXCEPT(AttributeDivestitureWasNotRequested)
RTI_EXCEPT(AttributeNotDefined)
RTI_EXCEPT(AttributeNotKnown)
RTI_EXCEPT(AttributeNotOwned)
RTI_EXCEPT(AttributeNotPublished)
RTI_EXCEPT(AttributeNotSubscribed)
RTI_EXCEPT(ConcurrentAccessAttempted)
RTI_EXCEPT(CouldNotDiscover)
RTI_EXCEPT(CouldNotOpenFED)
RTI_EXCEPT(CouldNotRestore)
RTI_EXCEPT>DeletePrivilegeNotHeld)
RTI_EXCEPT(DimensionNotDefined)
RTI_EXCEPT(EnableTimeConstrainedPending)
RTI_EXCEPT(EnableTimeConstrainedWasNotPending)
RTI_EXCEPT(EnableTimeRegulationPending)

```


RTI_EXCEPT(EnableTimeRegulationWasNotPending)
RTI_EXCEPT(ErrorReadingFED)
RTI_EXCEPT(EventNotKnown)
RTI_EXCEPT(FederateAlreadyExecutionMember)
RTI_EXCEPT(FederateDoesNotExist)
RTI_EXCEPT(FederateInternalError)
RTI_EXCEPT(FederateLoggingServiceCalls)
RTI_EXCEPT(FederateNotExecutionMember)
RTI_EXCEPT(FederateOwnsAttributes)
RTI_EXCEPT(FederateWasNotAskedToReleaseAttribute)
RTI_EXCEPT(FederatesCurrentlyJoined)
RTI_EXCEPT(FederationExecutionAlreadyExists)
RTI_EXCEPT(FederationExecutionDoesNotExist)
RTI_EXCEPT(FederationTimeAlreadyPassed)
RTI_EXCEPT(HandleValuePairMaximumExceeded)
RTI_EXCEPT(InteractionClassNotDefined)
RTI_EXCEPT(InteractionClassNotKnown)
RTI_EXCEPT(InteractionClassNotPublished)
RTI_EXCEPT(InteractionClassNotSubscribed)
RTI_EXCEPT(InteractionParameterNotDefined)
RTI_EXCEPT(InteractionParameterNotKnown)
RTI_EXCEPT(InvalidDivestitureCondition)
RTI_EXCEPT(InvalidExtents)
RTI_EXCEPT(InvalidFederationTime)
RTI_EXCEPT(InvalidLookahead)
RTI_EXCEPT(InvalidObjectHandle)
RTI_EXCEPT(InvalidOrderingHandle)
RTI_EXCEPT(InvalidOrderType)
RTI_EXCEPT(InvalidRegionContext)
RTI_EXCEPT(InvalidResignAction)
RTI_EXCEPT(InvalidRetractionHandle)
RTI_EXCEPT(InvalidTransportationHandle)
RTI_EXCEPT(InvalidTransportType)
RTI_EXCEPT(MemoryExhausted)
RTI_EXCEPT(NameNotFound)
RTI_EXCEPT(ObjectClassNotDefined)
RTI_EXCEPT(ObjectClassNotKnown)
RTI_EXCEPT(ObjectClassNotPublished)
RTI_EXCEPT(ObjectClassNotSubscribed)
RTI_EXCEPT(ObjectNotKnown)
RTI_EXCEPT(ObjectAlreadyRegistered)
RTI_EXCEPT(RegionNotKnown)
RTI_EXCEPT(RestoreInProgress)
RTI_EXCEPT(RestoreNotRequested)
RTI_EXCEPT(RTICannotRestore)
RTI_EXCEPT(RTIinternalError)
RTI_EXCEPT(SpaceNotDefined)
RTI_EXCEPT(SaveInProgress)
RTI_EXCEPT(SaveNotInitiated)
RTI_EXCEPT(SpecifiedSaveLabelDoesNotExist)
RTI_EXCEPT(SynchronizationPointLabelWasNotAnnounced)
RTI_EXCEPT(TimeAdvanceAlreadyInProgress)
RTI_EXCEPT(TimeAdvanceWasNotInProgress)
RTI_EXCEPT(TimeConstrainedAlreadyEnabled)
RTI_EXCEPT(TimeConstrainedWasNotEnabled)
RTI_EXCEPT(TimeRegulationAlreadyEnabled)
RTI_EXCEPT(TimeRegulationWasNotEnabled)
RTI_EXCEPT(UnableToPerformSave)
RTI_EXCEPT(UnimplementedService)
RTI_EXCEPT(UnknownLabel)

```
RTI_EXCEPT(ValueCountExceeded)
RTI_EXCEPT(ValueLengthExceeded)
```

```
enum TokenState {
    LOCKED = 1,
    UNLOCKED,
    DIVESTING,
    GONE
};
```

```
enum ObjectState {
    KNOWN_TO_FEDERATE = 1,
    HOLDING_TOKENS,
    DELETED
};
```

```
enum ResignAction {
    RELEASE_ATTRIBUTES = 1,
    DELETE_OBJECTS,
    DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES,
    NO_ACTION
};
```

```
enum FederateStateType {
    RUNNING = 1,
    SAVE_PENDING,
    SAVING,
    RESTORE_PENDING,
    RESTORING
};
```

```
enum TimeManagerStateType {
    IDLE = 1,
    TIME_ADVANCING
};
```

```
class Region;
```

```
class FederateAmbassador;
```

```
typedef FederateAmbassador *FederateAmbassadorPtr;
```

```
typedef RTI::Long SpaceHandle;
```

```
typedef RTI::ULong ObjectClassHandle;
```

```
typedef RTI::ULong InteractionClassHandle;
```

```
typedef RTI::ULong ExtentIndex;
```

```
typedef RTI::ULong Handle;
```

```
typedef Handle AttributeHandle;
```

```
typedef Handle ParameterHandle;
```

```
typedef Handle ObjectHandle;
```

```
typedef Handle DimensionHandle;
```

```
typedef RTI::ULong FederateHandle;

typedef Handle TransportationHandle;

typedef TransportationHandle TransportType;

typedef Handle OrderingHandle;
typedef OrderingHandle OrderType;

typedef RTI::ULong FederateID;
//typedef RTI::ULong ObjectID;

typedef RTI::ULong UniqueID;
//typedef RTI::UShort ObjectIDcount;

typedef RTI::Double FederationTimeDelta;
typedef RTI::Double FederationTime;
typedef RTI::Double TickTime;

// All char * declarations should use null terminated strings
typedef char * FederationExecutionName;
typedef char * FederationExecutionDataDesignator;
typedef char * FederateType;
typedef char * FEDsignature;
typedef char * FileName;
typedef char * SynchronizationLabel;
typedef char * SaveLabel;
typedef char * UserSuppliedTag;
typedef char * ObjectName;
typedef char * ObjectClassName;
typedef char * AttributeName;
typedef char * InteractionClassName;
typedef char * ParameterName;
typedef char * SpaceName;
typedef char * DimensionName;
typedef char * TransportationName;
typedef char * OrderingName;
```

```
// typedef HandleValuePairSet AttributeHandleValuePairSet;

class AttributeHandleValuePairSet {
// Instances of class HandleValuePairSet are the containers used to pass
// object attribute values and interaction parameter values between the
// Federate and the RTI. These containers hold sets of attribute/parameter
// values indexed by their attribute/parameter handle. Instances of this
// class are provided to the RTI in the Update Attribute Values and Send
// Interaction service invocations. Instances of this class are provided
// to the Federate in the Reflect Attribute Values and Receive Interaction
// service invocations. When instances of HandleValuePairSet are provided
// to the Federate by the RTI, the memory used to store attribute/parameter
// values is valid for use by the federate only within the scope of the
// Reflect Attribute Values or Receive Interaction service invocation.
// Symmetrically, for instances of HandleValuePairSet provided by the Federate
// to the RTI, the memory used to store attribute/parameter values is valid
for
// use by the RTI only within the scope of the Update Attribute Values or Send
// Interaction service invocation.
public:
    virtual ~AttributeHandleValuePairSet() { ; }

    virtual ULong size() const = 0;

    virtual Handle getHandle(
        RTI::ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual ULong getValueLength(
        RTI::ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void getValue(
        RTI::ULong i,
        char* buff,
        ULong& valueLength) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual char *getValuePointer(
        RTI::ULong i,
        ULong& valueLength) const
        throw (
            ArrayIndexOutOfBounds) = 0;

//    virtual RTI::TransportType getTransportType(
//        RTI::ULong i) const
//        throw (
//            ArrayIndexOutOfBounds) = 0;
//
//    virtual RTI::OrderType getOrderType(
//        RTI::ULong i) const
//        throw (
//            ArrayIndexOutOfBounds) = 0;
//
//    virtual RTI::Region getRegion(
```

```

//      RTI::ULong i) const
//      throw (
//          ArrayIndexOutOfBounds) = 0;

virtual void add(
    Handle      h,
    const char* buff,
    ULong       valueLength)
    throw (
        ValueLengthExceeded,
        ValueCountExceeded) = 0;

virtual void remove(           // not guaranteed safe while iterating
    Handle      h)
    throw (
        ArrayIndexOutOfBounds) = 0;

virtual void moveFrom(
    const AttributeHandleValuePairSet& ahvps,
    RTI::ULong& i)
    throw (
        ValueCountExceeded,
        ArrayIndexOutOfBounds) = 0;

virtual void empty() = 0; // Empty the Set without deallocating space.

virtual inline RTI::ULong start() const = 0;
virtual inline RTI::ULong valid(RTI::ULong i) const = 0;
virtual inline RTI::ULong next(RTI::ULong i) const = 0;
};

class AttributeSetFactory {
public:
    static AttributeHandleValuePairSet* create(
        ULong count)
        throw (
            MemoryExhausted,
            ValueCountExceeded,
            HandleValuePairMaximumExceeded);
};

class AttributeHandleSet {
public:
    virtual ~AttributeHandleSet() { ; }

    virtual ULong size() const = 0;

    virtual AttributeHandle getHandle(ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void add(AttributeHandle h)
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void remove(AttributeHandle h)
        throw (
            // not guaranteed safe while iterating
            AttributeNotDefined) = 0;
};

```

```

    virtual void empty() = 0; // Empty the Set

    virtual Boolean isEmpty() const = 0; //is set empty?
    virtual Boolean isMember(AttributeHandle h) const = 0;

// AttributeHandleSet setUnion(const AttributeHandleSet &) const;
// AttributeHandleSet setIntersection(const AttributeHandleSet &) const;
// AttributeHandleSet removeSetIntersection(const AttributeHandleSet &)
const;

};

class AttributeHandleSetFactory {
public:
    static AttributeHandleSet* create(
        ULong count)
        throw(
            MemoryExhausted,
            ValueCountExceeded);
};

class FederateHandleSet {
public:
    virtual ULong size() const = 0;

    virtual FederateHandle getHandle(ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void add(FederateHandle h)
        throw (
            ValueCountExceeded) = 0;

    virtual void remove(FederateHandle h)
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void empty() = 0; // Empty the set without deallocating space.
};

class FederateHandleSetFactory {
public:
    static FederateHandleSet* create(ULong count)
        throw (
            MemoryExhausted,
            ValueCountExceeded);
};

struct EventRetractionHandle_s {
    FederationTime theEventTime;
    UniqueID theSerialNumber;
    FederateHandle sendingFederate;
};
typedef struct EventRetractionHandle_s EventRetractionHandle;

// typedef HandleValuePairSet ParameterHandleValuePairSet;

class ParameterHandleValuePairSet {

```

```

// Instances of class HandleValuePairSet are the containers used to pass
// object attribute values and interaction parameter values between the
// Federate and the RTI. These containers hold sets of attribute/parameter
// values indexed by their attribute/parameter handle. Instances of this
// class are provided to the RTI in the Update Attribute Values and Send
// Interaction service invocations. Instances of this class are provided
// to the Federate in the Reflect Attribute Values and Receive Interaction
// service invocations. When instances of HandleValuePairSet are provided
// to the Federate by the RTI, the memory used to store attribute/parameter
// values is valid for use by the federate only within the scope of the
// Reflect Attribute Values or Receive Interaction service invocation.
// Symmetrically, for instances of HandleValuePairSet provided by the Federate
// to the RTI, the memory used to store attribute/parameter values is valid
for
// use by the RTI only within the scope of the Update Attribute Values or Send
// Interaction service invocation.
public:
    virtual ~ParameterHandleValuePairSet() { ; }

    virtual ULong size() const = 0;

    virtual Handle getHandle(
        RTI::ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual ULong getValueLength(
        RTI::ULong i) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void getValue(
        RTI::ULong i,
        char* buff,
        ULong& valueLength) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual char *getValuePointer(
        RTI::ULong i,
        ULong& valueLength) const
        throw (
            ArrayIndexOutOfBounds) = 0;

//    virtual RTI::TransportType getTransportType(void)
//        throw (
//            ArrayIndexOutOfBounds) = 0;

//    virtual RTI::OrderType getOrderType(void)
//        throw (
//            ArrayIndexOutOfBounds) = 0;

//    virtual RTI::Region getRegion(void)
//        throw (
//            ArrayIndexOutOfBounds) = 0;

    virtual void add(
        Handle h,
        const char* buff,
        ULong valueLength)

```

```

        throw (
            ValueLengthExceeded,
            ValueCountExceeded) = 0;

virtual void remove(           // not guaranteed safe while iterating
    Handle      h)
    throw (
        ArrayIndexOutOfBounds) = 0;

virtual void moveFrom(
    const ParameterHandleValuePairSet& phvps,
    RTI::ULong& i)
    throw (
        ValueCountExceeded,
        ArrayIndexOutOfBounds) = 0;

virtual void empty() = 0; // Empty the Set without deallocating space.

virtual inline RTI::ULong start() const = 0;
virtual inline RTI::ULong valid(RTI::ULong i) const = 0;
virtual inline RTI::ULong next(RTI::ULong i) const = 0;
};

class ParameterSetFactory {
public:
    static ParameterHandleValuePairSet* create(ULong count)
        throw (
            MemoryExhausted,
            ValueCountExceeded,
            HandleValuePairMaximumExceeded);
};

class Region {
public:

    virtual ~Region();

    virtual ULong getRangeLowerBound(
        ExtentIndex      theExtent,
        DimensionHandle theDimension) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual ULong getRangeUpperBound(
        ExtentIndex      theExtent,
        DimensionHandle theDimension) const
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void setRangeLowerBound(
        ExtentIndex      theExtent,
        DimensionHandle theDimension,
        ULong            theLowerBound)
        throw (
            ArrayIndexOutOfBounds) = 0;

    virtual void setRangeUpperBound(
        ExtentIndex      theExtent,
        DimensionHandle theDimension,

```



```
        ULong          theUpperBound)
    throw (
        ArrayIndexOutOfBounds) = 0;

virtual SpaceHandle getSpaceHandle() const
    throw (
        ) = 0;

virtual ULong getNumberOfExtents() const
    throw (
        ) = 0;

virtual ULong getRangeLowerBoundNotificationLimit(
    ExtentIndex      theExtent,
    DimensionHandle theDimension) const
    throw (
        ArrayIndexOutOfBounds) = 0;

virtual ULong getRangeUpperBoundNotificationLimit(
    ExtentIndex      theExtent,
    DimensionHandle theDimension) const
    throw (
        ArrayIndexOutOfBounds) = 0;
};
```

```

//File RTIambServices.hh
//Included in RTI.hh

//
//          RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by pointer value.
// C3  Function return by value.
// C4  In parameter by const pointer value.  Caller provides memory.
//      Caller may free memory or overwrite it upon completion of
//      the call.  Callee must copy during the call anything it
//      wishes to save beyond completion of the call.  Parameter
//      type must define const accessor methods.
// C5  Out parameter by pointer value.  Caller provides reference to object.
//      Callee constructs an instance on the heap (new) and returns.
//      The caller destroys the instance (delete) at its leisure.
// C6  Function return by pointer value.  Callee constructs an instance on
//      the heap (new) and returns a reference.  The caller destroys the
//      instance (delete) at its leisure.
//

typedef FederateAmbassador *FederateAmbassadorPtr;

////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

// 2.1
void createFederationExecution (
    const FederationExecutionName      executionName, // supplied C4
    const FederationExecutionDataDesignator FED)      // supplied C4
throw (
    FederationExecutionAlreadyExists,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 2.2
void destroyFederationExecution (
    const FederationExecutionName executionName) // supplied C4
throw (
    FederatesCurrentlyJoined,
    FederationExecutionDoesNotExist,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 2.3
FederateHandle                                     // returned C3
joinFederationExecution (
    const FederateType      yourName,                // supplied C4
    const FederationExecutionName executionName,      // supplied C4
    FederateAmbassadorPtr   federateAmbassadorReference) // supplied C1
throw (

```

```

    FederateAlreadyExecutionMember,
    FederationExecutionDoesNotExist,
    CouldNotOpenFED,
    ErrorReadingFED,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 2.4
void resignFederationExecution (
    ResignAction theAction) // supplied C1
throw (
    FederateOwnsAttributes,
    FederateNotExecutionMember,
    InvalidResignAction,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 2.5
void registerFederationSynchronizationPoint (
    const SynchronizationLabel label, // supplied C4
    const UserSuppliedTag theTag) // supplied C4
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

void registerFederationSynchronizationPoint (
    const SynchronizationLabel label, // supplied C4
    const UserSuppliedTag theTag, // supplied C4
    const FederateHandleSet& syncSet) // supplied C4
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 2.7
void synchronizationPointAchieved (
    const SynchronizationLabel label) // supplied C4
throw (
    SynchronizationPointLabelWasNotAnnounced,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 2.9
void requestFederationSave (
    const SaveLabel label, // supplied C4
    FederationTime theTime) // supplied C1
throw (
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,

```

```
        ConcurrentAccessAttempted,  
        SaveInProgress,  
        RestoreInProgress,  
        RTIinternalError);  
  
void requestFederationSave (  
    const SaveLabel label)      // supplied C4  
    throw (  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        SaveInProgress,  
        RestoreInProgress,  
        RTIinternalError);  
  
// 2.11  
void federateSaveBegun ()  
    throw (  
        SaveNotInitiated,  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        SaveInProgress,  
        RestoreInProgress,  
        RTIinternalError);  
  
// 2.12  
void federateSaveComplete ()  
    throw (  
        SaveNotInitiated,  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        RTIinternalError);  
  
void federateSaveNotComplete ()  
    throw (  
        SaveNotInitiated,  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        RTIinternalError);  
  
// 2.14  
void requestRestore (  
    const SaveLabel label) // supplied C4  
    throw (  
        SpecifiedSaveLabelDoesNotExist,  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        SaveInProgress,  
        RestoreInProgress,  
        RTIinternalError);  
  
// 2.17  
void restoreComplete ()  
    throw (  
        RestoreNotRequested,  
        RTICannotRestore,  
        FederateNotExecutionMember,  
        ConcurrentAccessAttempted,  
        RTIinternalError);  
  
void restoreNotComplete ()
```

```

throw (
    RestoreNotRequested,
    RTICannotRestore,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

// 3.1
void publishObjectClass (
    ObjectClassHandle theClass,          // supplied C1
    const AttributeHandleSet& attributeList) // supplied C4
throw (
    ObjectClassNotDefined,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.2
void unpublishObjectClass (
    ObjectClassHandle theClass) // supplied C1
throw (
    ObjectClassNotDefined,
    FederateOwnsAttributes,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.3
void publishInteractionClass (
    InteractionClassHandle theInteraction) // supplied C1
throw (
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.4
void unpublishInteractionClass (
    InteractionClassHandle theInteraction) // supplied C1
throw (
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.5
void subscribeObjectClassAttributes (

```

```

        ObjectClassHandle theClass,          // supplied C1
const AttributeHandleSet& attributeList, // supplied C4
        RTI::Boolean      active = RTI::RTI_TRUE)
throw (
    ObjectClassNotDefined,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.6
void unsubscribeObjectClass (
    ObjectClassHandle theClass) // supplied C1
throw (
    ObjectClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.7
void subscribeInteractionClass (
    InteractionClassHandle theClass, // supplied C1
    RTI::Boolean          active = RTI::RTI_TRUE)
throw (
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    FederateLoggingServiceCalls,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 3.8
void unsubscribeInteractionClass (
    InteractionClassHandle theClass) // supplied C1
throw (
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

////////////////////////////////////////
// Object Management Services //
////////////////////////////////////////

// 4.1
ObjectHandle          // returned C3
registerObjectInstance (
    ObjectClassHandle theClass, // supplied C1
    const ObjectName  theObject) // supplied C4
throw (
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    ObjectAlreadyRegistered,

```

```

    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

ObjectHandle                                // returned C3
registerObjectInstance (
    ObjectClassHandle theClass) // supplied C1
throw (
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 4.3
EventRetractionHandle                                // returned C3
updateAttributeValues (
    ObjectHandle theObject, // supplied C1
    const AttributeHandleValuePairSet& theAttributes, // supplied C4
    FederationTime theTime, // supplied C1
    const UserSuppliedTag theTag) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

void updateAttributeValues (
    ObjectHandle theObject, // supplied C1
    const AttributeHandleValuePairSet& theAttributes, // supplied C4
    const UserSuppliedTag theTag) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 4.5
EventRetractionHandle                                // returned C3
sendInteraction (
    InteractionClassHandle theInteraction, // supplied C1
    const ParameterHandleValuePairSet& theParameters, // supplied C4
    FederationTime theTime, // supplied C1
    const UserSuppliedTag theTag) // supplied C4
throw (
    InteractionClassNotDefined,
    InteractionClassNotPublished,

```

```

    InteractionParameterNotDefined,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

void sendInteraction (
    InteractionClassHandle    theInteraction, // supplied C1
    const ParameterHandleValuePairSet& theParameters, // supplied C4
    const UserSuppliedTag    theTag)        // supplied C4
throw (
    InteractionClassNotDefined,
    InteractionClassNotPublished,
    InteractionParameterNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 4.7
EventRetractionHandle // returned C3
deleteObjectInstance (
    ObjectHandle    ObjectHandle, // supplied C1
    FederationTime theTime,       // supplied C1
    const UserSuppliedTag theTag)  // supplied C4
throw (
    ObjectNotKnown,
    DeletePrivilegeNotHeld,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

void deleteObjectInstance (
    ObjectHandle    ObjectHandle, // supplied C1
    const UserSuppliedTag theTag) // supplied C4
throw (
    ObjectNotKnown,
    DeletePrivilegeNotHeld,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 4.9
void localDeleteObjectInstance (
    ObjectHandle    ObjectHandle) // supplied C1
throw (
    ObjectNotKnown,
    FederateOwnsAttributes,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,

```



```

        RTIinternalError);

// 4.10
void changeAttributeTransportType (
    ObjectHandle          theObject,          // supplied C1
    const AttributeHandleSet& theAttributes, // supplied C4
    TransportType         theType)           // Must be replaced by below
{
    ne.
    //      TransportationHandle      theHandle)           // supplied C1
    throw (
        ObjectNotKnown,
        AttributeNotDefined,
        AttributeNotOwned,
        InvalidTransportationHandle,
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

// 4.11
void changeInteractionTransportType (
    InteractionClassHandle theClass, // supplied C1
    TransportType          theType)  // Must be replaced by below line.
    // TransportationHandle theHandle) // supplied C1
{
    throw (
        InteractionClassNotDefined,
        InteractionClassNotPublished,
        InvalidTransportationHandle,
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

// 4.14
void requestObjectAttributeValueUpdate (
    ObjectHandle          theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
{
    throw (
        AttributeNotDefined,
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

void requestClassAttributeValueUpdate (
    ObjectClassHandle     theClass,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
{
    throw (
        ObjectClassNotDefined,
        AttributeNotDefined,
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

```

```

////////////////////////////////////
// Ownership Management Services //
////////////////////////////////////

// 5.1
void unconditionalAttributeOwnershipDivestiture (
    ObjectHandle          theObject,      // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.2
void negotiatedAttributeOwnershipDivestiture (
    ObjectHandle          theObject,      // supplied C1
    const AttributeHandleSet& theAttributes, // supplied C4
    const UserSuppliedTag theTag)         // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    AttributeAlreadyBeingDivested,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.6
void attributeOwnershipAcquisition (
    ObjectHandle          theObject,      // supplied C1
    const AttributeHandleSet& desiredAttributes, // supplied C4
    const UserSuppliedTag theTag)         // supplied C4
throw (
    ObjectNotKnown,
    ObjectClassNotPublished,
    AttributeNotDefined,
    AttributeNotPublished,
    FederateOwnsAttributes,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.7
void attributeOwnershipAcquisitionIfAvailable (
    ObjectHandle          theObject,      // supplied C1
    const AttributeHandleSet& desiredAttributes) // supplied C4
throw (
    ObjectNotKnown,
    ObjectClassNotPublished,
    AttributeNotDefined,
    AttributeNotPublished,

```

```

    FederateOwnsAttributes,
    AttributeAlreadyBeingAcquired,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.10
AttributeHandleSet*                                // returned C6
attributeOwnershipReleaseResponse (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    FederateWasNotAskedToReleaseAttribute,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.11
void cancelNegotiatedAttributeOwnershipDivestiture (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    AttributeDivestitureWasNotRequested,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.12
void cancelAttributeOwnershipAcquisition (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeAlreadyOwned,
    AttributeAcquisitionWasNotRequested,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIInternalError);

// 5.14
void queryAttributeOwnership (
    ObjectHandle      theObject,          // supplied C1
    AttributeHandle theAttribute) // supplied C1
throw (
    ObjectNotKnown,

```

```

    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 5.16
Boolean                                // returned C3
isAttributeOwnedByFederate (
    ObjectHandle    theObject,        // supplied C1
    AttributeHandle theAttribute)     // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

////////////////////////////////////
// Time Management Services //
////////////////////////////////////

// 6.1
void enableTimeRegulation (
    FederationTime    theFederateTime, // supplied C1
    FederationTimeDelta theLookahead)  // supplied C1
throw (
    FederateNotExecutionMember,
    TimeRegulationAlreadyEnabled,
    EnableTimeRegulationPending,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.3
void disableTimeRegulation ()
throw (
    FederateNotExecutionMember,
    TimeRegulationWasNotEnabled,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.4
void enableTimeConstrained ()
throw (
    FederateNotExecutionMember,
    TimeConstrainedAlreadyEnabled,
    EnableTimeConstrainedPending,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.6

```

```
void disableTimeConstrained ()
throw (
    FederateNotExecutionMember,
    TimeConstrainedWasNotEnabled,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.7
void timeAdvanceRequest (
    FederationTime theTime) // supplied C1
throw (
    TimeAdvanceAlreadyInProgress,
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.8
void timeAdvanceRequestAvailable (
    FederationTime theTime) // supplied C1
throw (
    TimeAdvanceAlreadyInProgress,
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.9
void nextEventRequest (
    FederationTime theTime) // supplied C1
throw (
    TimeAdvanceAlreadyInProgress,
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.10
void nextEventRequestAvailable (
    FederationTime theTime) // supplied C1
throw (
    TimeAdvanceAlreadyInProgress,
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);
```

```
// 6.11
void flushQueueRequest (
    FederationTime theTime) // supplied C1
throw (
    TimeAdvanceAlreadyInProgress,
    FederationTimeAlreadyPassed,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.13
void enableAsynchronousDelivery()
    throw (
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

// 6.14
void disableAsynchronousDelivery()
    throw (
        FederateNotExecutionMember,
        ConcurrentAccessAttempted,
        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

// 6.15
FederationTime // returned C3
queryLBTS ()
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.16
FederationTime // returned C3
queryFederateTime ()
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.17
FederationTime // returned C3
queryMinNextEventTime ()
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
```

```

    RTIinternalError);

// 6.18
void modifyLookahead (
    FederationTimeDelta theLookahead) // supplied C1
throw (
    InvalidLookahead,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.19
FederationTimeDelta // returned C3
queryLookahead ()
throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.20
void retract (
    EventRetractionHandle theHandle) // supplied C1
throw (
    InvalidRetractionHandle,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.22
void changeAttributeOrderType (
    ObjectHandle      theObject,      // supplied C1
    const AttributeHandleSet& theAttributes, // supplied C4
    OrderType         theType)        // Must be replaced by below line.
// OrderingHandle    theHandle)      // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    InvalidOrderingHandle,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 6.23
void changeInteractionOrderType (
    InteractionClassHandle theClass, // supplied C1
    OrderType              theType)  // Must be replaced by below line.
// OrderingHandle         theHandle) // supplied C1
throw (
    InteractionClassNotDefined,
    InteractionClassNotPublished,
    InvalidOrderingHandle,

```

```

    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

////////////////////////////////////
// Data Distribution Management //
////////////////////////////////////

// 7.1
Region*                                // returned C6
createRegion (
    SpaceHandle theSpace,    // supplied C1
    RTI::ULong  numberOfExtents) // supplied C1
throw (
    SpaceNotDefined,
    InvalidExtents,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.2
void notifyAboutRegionModification (
    Region &theRegion) // supplied C4
throw (
    RegionNotKnown,
    InvalidExtents,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.3
void deleteRegion (
    Region *theRegion) // supplied C1
throw (
    RegionNotKnown,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.4
ObjectHandle                                // returned C3
registerObjectInstanceWithRegion (
    ObjectClassHandle theClass,            // supplied C1
    const ObjectName  theObject,           // supplied C4
    AttributeHandle   theAttributes[],     // supplied C4
    Region            *theRegions[],       // supplied C4
    ULong             theNumberOfHandles) // supplied C1
throw (
    RegionNotKnown,
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    ObjectAlreadyRegistered,

```



```

    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

ObjectHandle                                     // returned C3
registerObjectInstanceWithRegion (
    ObjectClassHandle theClass,                // supplied C1
    AttributeHandle   theAttributes[],         // supplied C4
    Region             *theRegions[],         // supplied C4
    ULONG              theNumberOfHandles)     // supplied C1
throw (
    RegionNotKnown,
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.5
void associateRegionForUpdates (
    Region      &theRegion,    // supplied C4
    ObjectHandle theObject,    // supplied C1
    const AttributeHandleSet &theAttributes) // supplied C4
throw (
    RegionNotKnown,
    InvalidRegionContext,
    ObjectNotKnown,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.6
void unassociateRegionForUpdates (
    Region      &theRegion,    // supplied C4
    ObjectHandle theObject)     // supplied C1
throw (
    RegionNotKnown,
    InvalidRegionContext,
    ObjectNotKnown,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.7
void subscribeObjectClassAttributesWithRegion (
    ObjectClassHandle theClass,    // supplied C1
    Region             &theRegion, // supplied C4
    const AttributeHandleSet &attributeList, // supplied C4
    RTI::Boolean        active = RTI::RTI_TRUE)
throw (
    RegionNotKnown,

```

```

    ObjectClassNotDefined,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.8
void unsubscribeObjectClassWithRegion (
    ObjectClassHandle theClass,          // supplied C1
    Region            &theRegion)       // supplied C4
throw (
    RegionNotKnown,
    ObjectClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.9
void subscribeInteractionClassWithRegion (
    InteractionClassHandle theClass,      // supplied C1
    Region                &theRegion,    // supplied C4
    RTI::Boolean          active = RTI::RTI_TRUE)
throw (
    RegionNotKnown,
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    FederateLoggingServiceCalls,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.10
void unsubscribeInteractionClassWithRegion (
    InteractionClassHandle theClass,      // supplied C1
    Region                &theRegion)    // supplied C4
throw (
    RegionNotKnown,
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.11
EventRetractionHandle                // returned C3
sendInteractionWithRegion (
    InteractionClassHandle    theInteraction, // supplied C1
    const ParameterHandleValuePairSet &theParameters, // supplied C4
    FederationTime           theTime,         // supplied C1
    const UserSuppliedTag     theTag,         // supplied C4
    const Region              &theRegion)     // supplied C4
throw (
    RegionNotKnown,
    InteractionClassNotDefined,

```

```

    InteractionClassNotPublished,
    InteractionParameterNotDefined,
    InvalidFederationTime,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

void sendInteractionWithRegion (
    InteractionClassHandle    theInteraction, // supplied C1
    const ParameterHandleValuePairSet &theParameters, // supplied C4
    const UserSuppliedTag    theTag, // supplied C4
    const Region             &theRegion) // supplied C4
throw (
    RegionNotKnown,
    InteractionClassNotDefined,
    InteractionClassNotPublished,
    InteractionParameterNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 7.12
void requestClassAttributeValueUpdateWithRegion (
    ObjectClassHandle    theClass, // supplied C1
    const AttributeHandleSet &theAttributes, // supplied C4
    const Region         &theRegion) // supplied C4
throw (
    RegionNotKnown,
    ObjectClassNotDefined,
    AttributeNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

//////////
// RTI Support Services //
//////////

// 8.1
ObjectClassHandle // returned C3
getObjectClassHandle (
    const ObjectClassName theName) // supplied C4
throw (
    NameNotFound,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.2
ObjectClassName // returned C6
getObjectClassName (
    ObjectClassHandle theHandle) // supplied C1
throw (
    ObjectClassNotDefined,

```

```
FederateNotExecutionMember,  
ConcurrentAccessAttempted,  
RTIinternalError);  
  
// 8.3  
AttributeHandle // returned C3  
getAttributeHandle (  
    const AttributeName theName, // supplied C4  
    ObjectClassHandle whichClass) // supplied C1  
throw (  
    ObjectClassNotDefined,  
    NameNotFound,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.4  
AttributeName // returned C6  
getAttributeName (  
    AttributeHandle theHandle, // supplied C1  
    ObjectClassHandle whichClass) // supplied C1  
throw (  
    ObjectClassNotDefined,  
    AttributeNotDefined,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.5  
InteractionClassHandle // returned C3  
getInteractionClassHandle (  
    const InteractionClassName theName) // supplied C4  
throw (  
    NameNotFound,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.6  
InteractionClassName // returned C6  
getInteractionClassName (  
    InteractionClassHandle theHandle) // supplied C1  
throw (  
    InteractionClassNotDefined,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.7  
ParameterHandle // returned C3  
getParameterHandle (  
    const ParameterName theName, // supplied C4  
    InteractionClassHandle whichClass) // supplied C1  
throw (  
    InteractionClassNotDefined,  
    NameNotFound,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);
```

```
// 8.8
ParameterName // returned C6
getParameterName (
    ParameterHandle theHandle, // supplied C1
    InteractionClassHandle whichClass) // supplied C1
throw (
    InteractionClassNotDefined,
    InteractionParameterNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.9
ObjectHandle // returned C3
getObjectInstanceHandle (
    const ObjectName theName) // supplied C4
throw (
    NameNotFound,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.10
ObjectName // returned C6
getObjectInstanceName (
    ObjectHandle theHandle) // supplied C1
throw (
    ObjectNotKnown,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.11
SpaceHandle // returned C3
getRoutingSpaceHandle (
    const SpaceName theName) // supplied C4
throw (
    NameNotFound,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.12
SpaceName // returned C6
getRoutingSpaceName (
    const SpaceHandle theHandle) // supplied C4
throw (
    SpaceNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.13
DimensionHandle // returned C3
getDimensionHandle (
    const DimensionName theName, // supplied C4
    SpaceHandle whichSpace) // supplied C1
throw (
    SpaceNotDefined,
    NameNotFound,
```

```
FederateNotExecutionMember,  
ConcurrentAccessAttempted,  
RTIinternalError);  
  
// 8.14  
DimensionName // returned C6  
getDimensionName (  
    DimensionHandle theHandle, // supplied C1  
    SpaceHandle whichClass) // supplied C1  
throw (  
    SpaceNotDefined,  
    DimensionNotDefined,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.15  
SpaceHandle // returned C3  
getAttributeRoutingSpaceHandle (  
    AttributeHandle theHandle, // supplied C1  
    ObjectClassHandle whichClass) // supplied C1  
throw (  
    ObjectClassNotDefined,  
    AttributeNotDefined,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.16  
ObjectClassHandle // returned C3  
getObjectClass (  
    ObjectHandle theObject) // supplied C1  
throw (  
    ObjectNotKnown,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.17  
SpaceHandle // returned C3  
getInteractionRoutingSpaceHandle (  
    InteractionClassHandle theHandle) // supplied C1  
throw (  
    InteractionClassNotDefined,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.18  
TransportationHandle // returned C3  
getTransportationHandle (  
    const TransportationName theName) // supplied C4  
throw (  
    NameNotFound,  
    FederateNotExecutionMember,  
    ConcurrentAccessAttempted,  
    RTIinternalError);  
  
// 8.19  
TransportationName // returned C6
```

```
getTransportationName (
    TransportationHandle theHandle) // supplied C1
throw (
    InvalidTransportationHandle,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.20
OrderingHandle // returned C3
getOrderingHandle (
    const OrderingName theName) // supplied C4
throw (
    NameNotFound,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.21
OrderingName // returned C6
getOrderingName (
    OrderingHandle theHandle) // supplied C1
throw (
    InvalidOrderingHandle,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 8.22
void enableClassRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.23
void disableClassRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.24
void enableAttributeRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.25
void disableAttributeRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
```

```

        SaveInProgress,
        RestoreInProgress,
        RTIinternalError);

// 8.26
void enableAttributeScopeAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.27
void disableAttributeScopeAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.28
void enableInteractionRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.29
void disableInteractionRelevanceAdvisorySwitch()
throw(
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

//
Boolean // returned C3
tick ()
throw (
    SpecifiedSaveLabelDoesNotExist,
    ConcurrentAccessAttempted,
    RTIinternalError);

Boolean // returned C3
tick (
    TickTime minimum, // supplied C1
    TickTime maximum) // supplied C1
throw (
    SpecifiedSaveLabelDoesNotExist,
    ConcurrentAccessAttempted,
    RTIinternalError);

RTIambassador()
throw (
    MemoryExhausted,

```



```
        RTIinternalError);

~RTIambassador()
throw (RTIinternalError);

// 8.XXX
ULong                                     // returned C6
getNumberOfDimensions (
    const SpaceHandle theHandle) // supplied C4
throw (
    SpaceNotDefined,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);
```

```

//File federateAmbServices.hh
//Included in RTI.hh

//
//          RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by pointer value.
// C3  Function return by value.
// C4  In parameter by const pointer value.  Caller provides memory.
//      Caller may free memory or overwrite it upon completion of
//      the call.  Callee must copy during the call anything it
//      wishes to save beyond completion of the call.  Parameter
//      type must define const accessor methods.
// C5  Out parameter by pointer value.  Caller provides reference to object.
//      Callee constructs an instance on the heap (new) and returns.
//      The caller destroys the instance (delete) at its leisure.
// C6  Function return by pointer value.  Callee constructs an instance on
//      the heap (new) and returns a reference.  The caller destroys the
//      instance (delete) at its leisure.
//
//
//
//
//
// Federation Management Services //
//
//
// 2.?
virtual void synchronizationPointRegistrationSucceeded (
    const SynchronizationLabel label) // supplied C4)
throw (
    FederateInternalError) = 0;

virtual void synchronizationPointRegistrationFailed (
    const SynchronizationLabel label) // supplied C4)
throw (
    FederateInternalError) = 0;

// 2.6
virtual void announceSynchronizationPoint (
    const SynchronizationLabel label, // supplied C4
    const UserSuppliedTag tag) // supplied C4
throw (
    FederateInternalError) = 0;

// 2.8
virtual void federationSynchronized (
    const SynchronizationLabel label) // supplied C4)
throw (
    FederateInternalError) = 0;

// 2.10
virtual void initiateFederateSave (
    const SaveLabel label) // supplied C4
throw (
    UnableToPerformSave,
    FederateInternalError) = 0;

// 2.13

```

```

virtual void federationSaved ()
throw (
    FederateInternalError) = 0;

virtual void federationNotSaved ()
throw (
    FederateInternalError) = 0;

// 2.15
virtual void federationSaveBegun ()
throw (
    FederateInternalError) = 0;

// 2.16
virtual void initiateRestore (
    const SaveLabel      label,    // supplied C4
    FederateHandle handle) // supplied C1
throw (
    SpecifiedSaveLabelDoesNotExist,
    CouldNotRestore,
    FederateInternalError) = 0;

// 2.18
virtual void federationRestored ()
throw (
    FederateInternalError) = 0;

virtual void federationNotRestored ()
throw (
    FederateInternalError) = 0;

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

// 3.9
virtual void startRegistrationForObjectClass (
    ObjectClassHandle theClass) // supplied C1
throw (
    ObjectClassNotPublished,
    AttributeNotPublished,
    FederateInternalError) = 0;

// 3.10
virtual void stopRegistrationForObjectClass (
    ObjectClassHandle theClass) // supplied C1
throw (
    ObjectClassNotPublished,
    AttributeNotPublished,
    FederateInternalError) = 0;

// 3.11
virtual void turnInteractionsOn (
    InteractionClassHandle theHandle) // supplied C1
throw (
    InteractionClassNotPublished,
    FederateInternalError) = 0;

// 3.12
virtual void turnInteractionsOff (

```

```

    InteractionClassHandle theHandle) // supplied C1
throw (
    InteractionClassNotPublished,
    FederateInternalError) = 0;

//////////
// Object Management Services //
//////////

// 4.2
virtual void discoverObjectInstance (
    ObjectHandle      theObject,      // supplied C1
    ObjectClassHandle theObjectClass) // supplied C1
throw (
    CouldNotDiscover,
    ObjectClassNotKnown,
    FederateInternalError) = 0;

// 4.4
virtual void reflectAttributeValues (
    ObjectHandle      theObject,      // supplied C1
    const AttributeHandleValuePairSet& theAttributes, // supplied C4
    FederationTime    theTime,        // supplied C1
    const UserSuppliedTag theTag,      // supplied C4
    EventRetractionHandle theHandle)  // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    InvalidFederationTime,
    FederateInternalError) = 0;

virtual void reflectAttributeValues (
    ObjectHandle      theObject,      // supplied C1
    const AttributeHandleValuePairSet& theAttributes, // supplied C4
    const UserSuppliedTag theTag)      // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

// 4.6
virtual void receiveInteraction (
    InteractionClassHandle theInteraction, // supplied C1
    const ParameterHandleValuePairSet& theParameters, // supplied C4
    FederationTime    theTime,        // supplied C1
    const UserSuppliedTag theTag,      // supplied C4
    EventRetractionHandle theHandle)  // supplied C1
throw (
    InteractionClassNotKnown,
    InteractionParameterNotKnown,
    InvalidFederationTime,
    FederateInternalError) = 0;

virtual void receiveInteraction (
    InteractionClassHandle theInteraction, // supplied C1
    const ParameterHandleValuePairSet& theParameters, // supplied C4
    const UserSuppliedTag theTag)      // supplied C4
throw (
    InteractionClassNotKnown,
    InteractionParameterNotKnown,

```

```
FederateInternalError) = 0;

// 4.8
virtual void removeObjectInstance (
    ObjectHandle      theObject, // supplied C1
    FederationTime    theTime,   // supplied C1
    const UserSuppliedTag theTag, // supplied C4
    EventRetractionHandle theHandle) // supplied C1
throw (
    ObjectNotKnown,
    InvalidFederationTime,
    FederateInternalError) = 0;

virtual void removeObjectInstance (
    ObjectHandle      theObject, // supplied C1
    const UserSuppliedTag theTag) // supplied C4
throw (
    ObjectNotKnown,
    FederateInternalError) = 0;

// 4.12
virtual void attributesInScope (
    ObjectHandle      theObject, // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

// 4.13
virtual void attributesOutOfScope (
    ObjectHandle      theObject, // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

// 4.15
virtual void provideAttributeValueUpdate (
    ObjectHandle      theObject, // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

// 4.16
virtual void turnUpdatesOnForObjectInstance (
    ObjectHandle      theObject, // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotOwned,
    FederateInternalError) = 0;

// 4.17
virtual void turnUpdatesOffForObjectInstance (
    ObjectHandle      theObject, // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
```

```

throw (
    ObjectNotKnown,
    AttributeNotOwned,
    FederateInternalError) = 0;

////////////////////////////////////
// Ownership Management Services //
////////////////////////////////////

// 5.3
virtual void requestAttributeOwnershipAssumption (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& offeredAttributes, // supplied C4
    const UserSuppliedTag   theTag)       // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAlreadyOwned,
    AttributeNotPublished,
    FederateInternalError) = 0;

// 5.4
virtual void attributeOwnershipDivestitureNotification (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& releasedAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeNotOwned,
    AttributeDivestitureWasNotRequested,
    FederateInternalError) = 0;

// 5.5
virtual void attributeOwnershipAcquisitionNotification (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& securedAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAcquisitionWasNotRequested,
    AttributeAlreadyOwned,
    AttributeNotPublished,
    FederateInternalError) = 0;

// 5.8
virtual void attributeOwnershipUnavailable (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAlreadyOwned,
    AttributeAcquisitionWasNotRequested,
    FederateInternalError) = 0;

// 5.9
virtual void requestAttributeOwnershipRelease (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& candidateAttributes, // supplied C4
    const UserSuppliedTag   theTag)       // supplied C4

```

```

throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeNotOwned,
    FederateInternalError) = 0;

// 5.13
virtual void confirmAttributeOwnershipAcquisitionCancellation (
    ObjectHandle      theObject,          // supplied C1
    const AttributeHandleSet& theAttributes) // supplied C4
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAlreadyOwned,
    AttributeAcquisitionWasNotCanceled,
    FederateInternalError) = 0;

// 5.15
virtual void informAttributeOwnership (
    ObjectHandle      theObject,          // supplied C1
    AttributeHandle theAttribute, // supplied C1
    FederateHandle theOwner)           // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

virtual void attributeIsNotOwned (
    ObjectHandle      theObject,          // supplied C1
    AttributeHandle theAttribute) // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

virtual void attributeOwnedByRTI (
    ObjectHandle      theObject,          // supplied C1
    AttributeHandle theAttribute) // supplied C1
throw (
    ObjectNotKnown,
    AttributeNotKnown,
    FederateInternalError) = 0;

////////////////////////////////////
// Time Management Services //
////////////////////////////////////

// 6.2
virtual void timeRegulationEnabled (
    FederationTime theFederateTime) // supplied C1
throw (
    InvalidFederationTime,
    EnableTimeRegulationWasNotPending,
    FederateInternalError) = 0;

// 6.5
virtual void timeConstrainedEnabled (
    FederationTime theFederateTime) // supplied C1
throw (
    InvalidFederationTime,

```

```
    EnableTimeConstrainedWasNotPending,  
    FederateInternalError) = 0;  
  
// 6.12  
virtual void timeAdvanceGrant (  
    FederationTime theTime) // supplied C1  
throw (  
    InvalidFederationTime,  
    TimeAdvanceWasNotInProgress,  
    FederationTimeAlreadyPassed,  
    FederateInternalError) = 0;  
  
// 6.21  
virtual void requestRetraction (  
    EventRetractionHandle theHandle) // supplied C1  
throw (  
    EventNotKnown,  
    FederateInternalError) = 0;
```


ANNEX C (normative)

Ada 95 application programmer's interface

TBS

ANNEX D (normative)

Java application programmer's interface

TBS

ANNEX E (informative)

Bibliography

[A1] Harel, David. "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming* (Netherlands) 8, 3 (June 1987): 231-274.

[A2] U.S. Department of Defense, *High Level Architecture, Data Distribution Management: Design Document Version 0.5*, February 1997.

[A3] U.S. Department of Defense, "Annotated Briefing on the DoD High Level Architecture for Simulation." Briefing, Defense Modeling and Simulation Office. Available WWW:
<URL: <http://hla.dmsso.mil/hla/general/annotate/sld001.htm>>

[A4] U.S. Department of Defense, *High Level Architecture, Time Management: Design Document*, Defense Modeling and Simulation Office. Available WWW:
<URL: http://hla.dmsso.mil/hla/tech/ifspect/TIM_MGT1.DOC>